IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	:	
	:	
Shotton et al.	:	Group Art Unit: 2127
	:	
Appln. No.: 09/615,830	:	
	:	Examiner: L. A. BULLOCK
Filed: July 13, 2000	:	
	:	
For: Locally Executing Software Agent for Retrieving Remote Content and Methods for Creation and Use of the Agent	:	

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF PRIOR INVENTION UNDER 37 C.F.R. § 1.131

Sir:

I, Charles T. Shotton, Jr., hereby declare that I am a co-inventor of the invention that is described and claimed in the above-identified patent application. I also hereby declare that prior to May 5, 1999, my co-inventors and I conceived of and reduced to practice the invention that is described and claimed in the above-identified patent application, as evidenced by the following:

1. Prior to May 5, 1999, my co-inventors and I conceived of the invention in the United States of America.

2. Shortly after my co-inventors and I conceived of the invention, and prior to May 5, 1999, my co-inventors and I prepared an Executive Summary of the invention. A date-redacted copy of the "Gossip Executive Summary" of the invention is attached hereto as Exhibit A.

3. Shortly after preparing the Executive Summary of the invention, and prior to May 5, 1999, my co-inventors and I prepared an overview of the architecture of an embodiment of the invention. A date-redacted copy of the "Gossip Architecture Overview, LOE, and Staffing Requirements" is attached hereto as Exhibit B.

4. Prior to May 5, 1999, my co-inventors and I reduced the invention to practice. A date-redacted printout of several source code files is attached hereto as Exhibit C.

5. Prior to May 5, 1999, my co-inventors and I prepared a diagram of the architecture of a reduction to practice of the invention. A date-redacted copy of the architecture diagram is attached hereto as Exhibit D.

6. Prior to May 5, 1999, my co-inventors and I used a reduction to practice of the invention to form a retrieval-content web page. A date-redacted printout of a screenshot of the retrieved-content web page is attached hereto as Exhibit E.

I further hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

DECLARANT: _____
Charles T. Shotton, Jr.

Date: 4/15/2005

EXHIBIT A

Gossip Executive Summary

BIAP Systems, Inc.

Introduction

"Gossip" is the code word for a new software system being developed by BIAP Systems, Inc. Gossip is intended to change the way individuals interact with the Internet and their local intranet by radically increasing their ability to sort through large amounts of information arriving via e-mail, the Web, and other desktop applications. In addition, the Gossip system introduces an new genre of personal information collaboration, allowing individuals to search for and exchange trusted, high quality data with others sharing their interests.

The product works with existing Internet standards, software such as Netscape Navigator or Microsoft Internet Explorer, HTTP, e-mail, and file transfer protocols, and provides a level of intelligence to help manage the high volume of information traffic flowing through an individual's workstation. In short, it makes your 'Net connection smarter.

Customer Need

The information management skills of the average Internet user are overtaxed. Users are forced to actively navigate the Web, manually search for information in their areas of interest, and to resort to traditional collaboration mechanisms like e-mail and on-line chat to communicate with peers. Information discovery via traditional search engines is cumbersome and the search results are of dubious quality. Information management through the use of bookmarks, Post-Its stuck to monitors, and paper print-outs clearly illustrates the need for a new technology that adds a layer of automation to the Web surfing experience, providing transparent information management tools for novice as well as experienced users.

Attempts at automating collaboration, information discovery, and information organization have focused on quick-to-market solutions that require the user to take an active role in the decision making process associated with every item of data. With the increased information flow through the average desktop PC, users no longer want or can afford to manage every single data element that passes through their workstation. Intelligent filtering of e-mail, automated information retrieval from high quality sources, and agent-based interactions with other users and sites on the Internet are much more desirable mechanisms for coping with the flood of data.

Existing Products

Several recent attempts have been made to address some of the problems associated with organizing, managing, and distributing the high volume of information confronting the average user. Collaborative filtering efforts like WiseWire and FireFly attempt to resolve the information discovery problem by centralizing a repository of subscriber submitted links, their qualitative assessments of the associated data, and

some pattern matching mechanisms for associating those data with user queries. Flaws with this approach include dependence on a large, centralized data store, reliance on qualitative measures from unknown sources, and an inability to be tailored to a specific user's work habits. The result is a somewhat homogeneous information base with limited subject areas that forces users to adopt information organization conventions imposed by the central site.

Publishing and exchanging information in a collaborative fashion has also been attempted through the creation of new "pseudo-push" services from companies like Marimba, BackWeb, and PointCast. Again, the dependence on a centralized "transmitter" site with limited facilities for customization by the end user restrict the long-term value of these services and products.

The Gossip Concept

The Gossip system is engineered to solve three problems confronting users of networked computers. With the exponential growth of the Internet, the accompanying increase in e-mail traffic and Web-based information has become unmanageable for the average computer users. Gossip provides agent-based technologies that monitor the information flow, make intelligent decisions about categorizing and prioritizing the information flow, and learn from user interactions with traditional network applications like e-mail, news, and Web browsers.

Gossip also makes information discovery and exchange simpler and more reliable. It provides mechanisms based on standard Internet protocols that allow users to perform distributed searches across Gossip systems. This information is based on the categorizations and quality assessments performed by other trusted human users and Gossip agents as opposed to brute force keyword searches as provided by search engines like InfoSeek or Excite.

Finally, Gossip automates network-based communications, providing a vehicle for both novice and experienced users to develop real-time collaborative communications applications, communities of interest, and commerce based solutions. Gossip's intelligent scheduling system allows automated publishing and retrieval operations which implement true "push" functionality.

Gossip Markets

The Gossip system represents a scalable information infrastructure that is customized for three target markets. A low-end consumer version fills the need for the home or hobbyist user and helps accelerate market adoption. A high-end version provides information management tools for use in intranet environments. A third service component that implements a centralized information exchange creates an annuity stream that supplements revenue from sales of the two software products in traditional distribution channels.

Because Gossip represents a new enabling infrastructure for Internet users, rapid, widespread adoption is desirable. This calls for the implementation of a two-tiered market penetration strategy, with free and low-cost consumer versions distributed electronically and custom, high value products marketed through direct sales and OEM agreements. The service component provides revenue generation from all customers, including users of a low-end or free product.

Window of Opportunity

The trade press is just beginning to recognize the need to manage increased information flow. Current reviews of browser based products are critical of the low level of automation in these applications. Likewise, on-line search engines are lagging behind the need for advanced information discovery tools. BIAP believes that a product like Gossip, if introduced in the next 9 to 12 months, will be uniquely positioned to answer end user needs for enhanced information management tools. Our past experience with delivering Internet based products as the wave of media interest in them crests shows that the time for this product is within the next year.

Market Value

Based on the demand for its previous products, BIAP believes the market for a consumer level version of the Gossip product conservatively approaches 100,000 units in the first 12 months of distribution. With a sub-$100 price point, it is anticipated that the initial development costs of the product can be recovered within the first 6 months of operation. Valuation of the product and BIAP Systems, Inc. vary, based on the exit strategies discussed below. For a traditional 3 year start-up, we estimate the valuation of the company to be between $35M and $60M.

If Gossip productization is completed and the resulting products are licensed or sold within the first 6 to 12 months, the one-time sale price is estimated at between $10M and $30M, depending on the amount of market penetration and media attention prior to the sale. This is on par with other start-up software products such as Ceneca's PageMill and SiteMill (sold to Adobe for $15M) which were developed and licensed before a widespread market was established.

PRODUCT STRATEGY

Vision and Architecture

Gossip is designed to provide a powerful, yet unobtrusive set of tools for making interaction with networked information services easier. The ideal marketing message is that "Gossip makes your computer work the way they do in the movies." The product consists of four major architecture components. They are the object repository, the expert system, the network interfaces, and the agents. In a deployed system, each user has their own Gossip installation, with local object store, customized expert system and knowledge base, network interfaces tuned to their information retrieval needs, and agents tailored to their work patterns.

Gossip is implemented in a combination of Java and ANSI Standard C. This allows the entire Gossip suite to be quickly ported to new platforms. The initial Gossip implementation supports both Windows 95/NT and Macintosh platforms. For performance reasons, the object repository and portions of the expert system are implemented in C. The repository provides a persistent storage mechanism for real-world entities like information about people, Web sites, documents, scheduled events, other Gossip locations on the Internet, and security and trust information. The expert system manipulates information in the object repository, making decisions about how to categorize information, when to rebroadcast it, whether the user should be notified about new information, etc.

Agents are invoked to perform specific functions on information in the object repository. Custom applications may be developed and integrated as Gossip agents, allowing the entire Gossip system to grow without limits as new applications, agents, and communications mechanisms are developed. Agents can be invoked in response to incoming information, actions taken by a user, or as part of learned behavior by the expert system. Agents are the primary consumer of services from the network interface component, which provides a transparent link between traditional client applications like Web browsers, e-mail clients, and news readers and the rest of the Internet and its services. Information flowing in and out of these more traditional tools flows through the Gossip system, allowing the expert system and agents to operate on the data transparently to the user.

Gossip users can interact with the system through their normal Internet client applications, or they can use customized interfaces provided by the agent software. The following sections describe a few of the typical interactions Gossip users may have with their system.

Operational Scenarios

The simplest scenario is that of a stand-alone user relying on Gossip to manage information discovery during Web surfing. Consider a user that has defined several

broad categories of information they are interested, such as sports, investments, and entertainment. As a normal course of operation, Gossip tracks all interactions the user has with a Web browser. Perhaps the user encounters a Web page with information on the Cubs and would like to save it for future reference. Gossip is already aware of the location of the page on the Internet, the content of the page, and has already performed a preliminary analysis of the content. By detecting terms like "baseball", "score", and "pitcher", the expert system will make a preliminary determination that the document probably belongs in the "Sports" category. The user is asked for confirmation that this decision is correct, and the document and all associated information are archived in the object repository.

If the expert system has mistakenly assumed an incorrect category, the user is given the option to override the decision and can take the opportunity to teach the system about the proper categorization. After only a few training sessions, the Gossip system is able to recognize and properly categorize most information without user intervention. Once information is stored in the object repository, the user may search and query the information as with any on-line information retrieval tool.

The real power of the Gossip system is its ability to exchange, search and retrieve information with other Gossip users. Suppose the user decides to create a sub-category of Sports, called "Baseball info for friends." The user can define destinations for this information, informing the system that any data categorized as "Baseball info for friends" should be forwarded automatically to a selected group of individuals (actually their Gossip systems) that the system is aware of. Simply surfing the net and saving baseball information in this new category automatically notifies the remote systems that the user has found information that they may be interested in. By establishing trust and reliability levels for individuals known to the system, information discovery reaches a much higher level of accuracy than traditional brute force search engines.

This "gossiping" metaphor allows users to easily establish communities of interest on any subject, exchange information in a highly automated fashion, and allows users to access a reliable, trusted base of information in a particular subject area. A user may request that his Gossip system find the most recent information about a particular subject. If the information isn't present in the local repository and the agents that the user has installed cannot find the appropriate information on the network at large, the Gossip system may choose to ask other Gossip systems for help with the query. If the user has defined a set of trusted friends, co-workers, or other known sites running the Gossip application, the query can be forwarded to the appropriate individuals' Gossip systems. If those systems cannot produce a reliable result, the users themselves may be prompted to help produce a solution, or their own object repositories may know of other Gossip systems further afield that may have the requested information. Much as a person in an office might walk down the hall asking co-workers for information on a specific subject, the Gossip applications automate this process, "gossiping" amongst themselves until an answer is found or all avenues are exhausted. Because repositories

generally contain refined information that is well-categorized, search results are more focused and considerably more accurate.

This categorization and information exchange process works equally well for incoming e-mail messages. After learning a user's habits for categorizing and accessing e-mail, the Gossip system can automatically filter e-mail, discarding obvious junk mail, prioritizing important messages, and archiving all messages. Because the system supports various levels of security and access controls, a user doesn't have to worry about a Gossip request inadvertently returning private e-mail or other information to another user. The system implements an "instant messaging" function, allowing users to be notified of urgent communications, search results, or messages from other Gossip systems.

The built-in scheduling function also allows the system to be used as a personal assistant to organize information and present it at the appropriate time. For instance, the user may inform the Gossip system that on Tuesday afternoon at 3 PM, he has a meeting with Bob and Mary regarding Project X. At 2:30, the Gossip system could be configured to send an instant message to Bob and Mary as a reminder, open up the latest Project X specifications in the user's Web browser, and query Bob and Mary's Gossip systems for the latest information that they have on the project.

Because the system has a rich set of agents for performing actions like searches, notifications, and other network operations, it is very easy to construct quite elaborate applications from a simple set of tools. In most cases, users can operate the system by simply answering yes/no questions posed by the expert system. For more detailed scenarios, the user can use straightforward applications to define a series of operations to happen in response to an event (like an incoming message from a specific source, or a certain elapsed time) and associate it with the appropriate data items.

Development Status

The Gossip product development effort was begun in [redacted] and is currently in development by BIAP Systems, Inc. The development team consists of 6 senior individuals with more than 75 years combined experience in software development and program management. The project is currently being funded by the team members and there are no outside investors at this time. Completion of an initial product is scheduled for [redacted] with widespread distribution beginning in [redacted]

Software Release Options

There are three options for getting the Gossip product to market. The traditional approach is to grow the team to approximately 25 members over the next 9 months, relying on electronic sales of the product to bootstrap sales into more traditional channels. With appropriate funding, the company can accelerate market penetration and grow at a much faster pace.

The second option is to complete product development, market the completed product to large software houses, and enter into an exclusive licensing arrangement for a third party to market and distribute the product. This allows the original team to begin development on follow-on products and avoids the need to ramp up to a full-blown corporate structure for supporting the complete product lifecycle. This model has been successfully followed by companies such as Spyglass.

The final option is to complete the product, market to interested third parties, and sell the intellectual property outright. This provides the quickest exit strategy and a reasonable return on initial investments. As mentioned, Ceneca followed this model with their PageMill product, selling to Adobe.

Development and Funding Timeline

With a planned ship date of [redacted] the Gossip team must accelerate the development process. While the project is on track, increased staffing requirements and time to market considerations require that BIAP seek additional funding. $1.5M to $3M in funding is being sought to increase management, marketing, and engineering staff and to begin marketing activities prior to product release. First year revenues of approximately $4M are projected. It is anticipated that an initial round of funding at this level is sufficient to allow the product to bootstrap itself beyond the first year.

Key Personnel

Chuck Shotton - Mr. Shotton is the founder of BIAP Systems, Inc. He has 14+ years of software engineering and project management experience, including positions of Sr. VP of Engineering and Fellow for Quarterdeck Corporation.

Louis Slothouber, Ph.D. - Dr. Slothouber's primary areas of expertise include artificial intelligence, language processing, and production system design. Previous positions include professor of computer science for the University of Houston and Computer Scientist for Quarterdeck Corporation.

M. Ellen Dudar - Ms. Dudar brings 12 years of software engineering and design to the Gossip project. Her background include project management and software development for the Space Station Freedom project and 5 years as a consulting engineer for Oracle Corporation, specializing in database design and performance.

Christine Smith - Ms. Smith has 16 years of software development expertise to contribute to the Gossip effort. Her background includes project management and software engineering efforts for the Space Shuttle program, several military flight simulator systems, and human factors and user interface design experience.

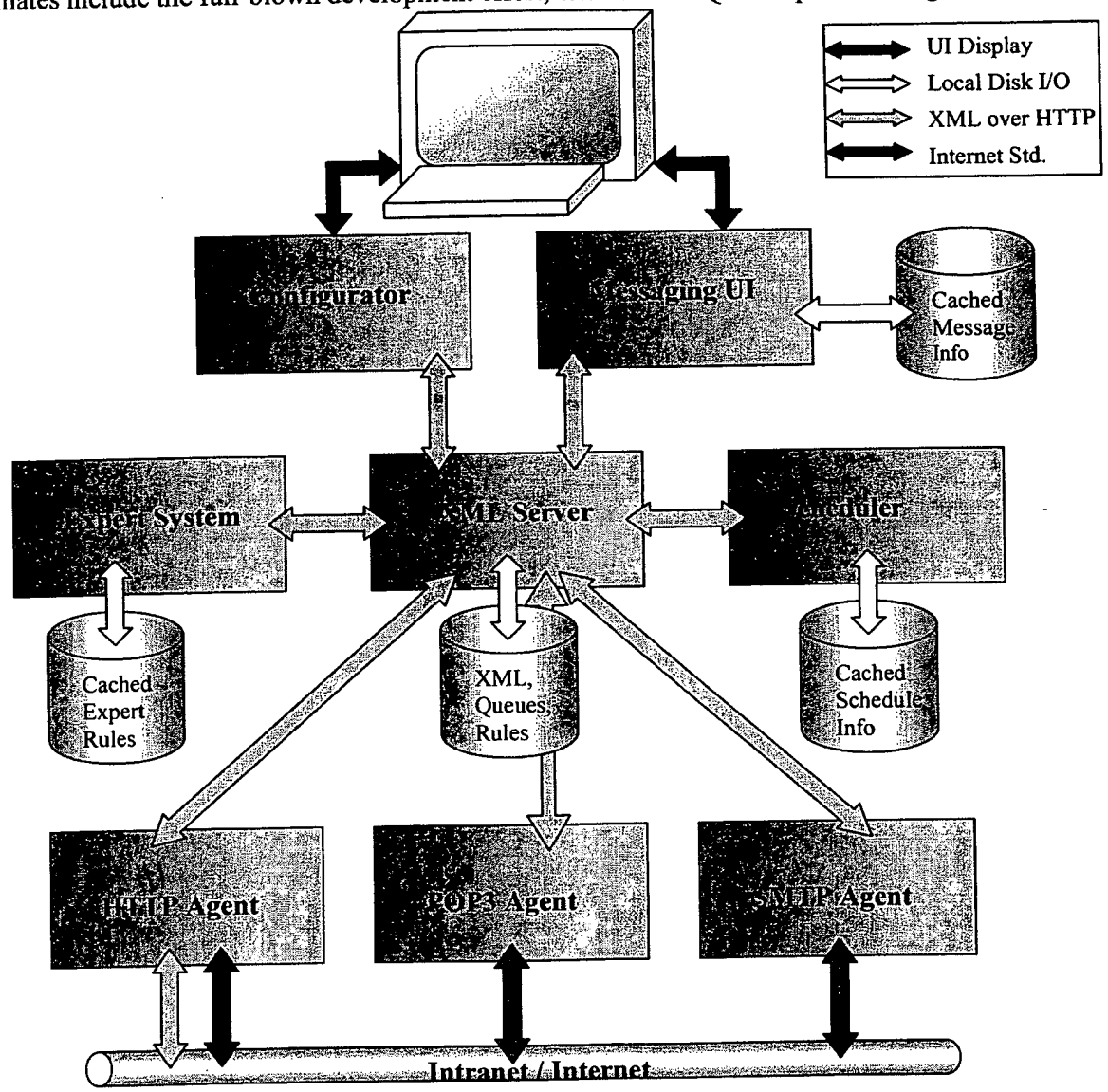
Linda Shotton - Ms. Shotton has been involved with BIAP's earlier networking products, providing project management and software engineering support. Her background includes 10+ years of software design and engineering experience on DoD, NASA, and civilian information systems and flight simulation projects.

Todd Nix - Mr. Nix provides 11 years of software engineering experience, including participation in several start-up ventures where he was chief architect for financial and investment software systems. Recent experience includes software development and engineering for Dell Computer and work on the Space Station Freedom's Software Support Environment.

EXHIBIT B

Gossip Architecture Overview, LOE, and Staffing Requirements

This document describes the high level architectural elements of the Gossip system and describes staffing, schedule, and design considerations. This document is a moving target and staffing and schedule considerations may change over time. The diagram below shows the basic Gossip components and how they are integrated. The assumption is that the Gossip system can exist on a single desktop CPU or can be scaled to span multiple networked computers in a network infrastructure. What follows is a description of each architectural element, design assumptions, and staffing and schedule requirements. Man-month estimates include the full-blown development effort, exclusive of QA or up front design considerations.



The staffing requirements indicate the skill level required for completion of the task in the time specified and are predicated on the assumption that the staff members performing the work are integrated with the team and up to speed on the system architecture and design and implementation goals.

XML Server

Description:

The XML Server is the central element in the Gossip architecture. It functions as a storage repository for messages between other elements, rules and configuration information for elements, and knowledge and data captured from the user. It also holds persistent versions of queues that are used to drive other processes. It provides the ability to retrieve, modify, and delete XML objects by name, provides exclusive or shared access to XML objects, provides a mechanism for notifying other elements when specific XML objects change.

An invisible companion portion of this element is a set of library routines for parsing XML served up by the XML server as well as the associated HTTP client code that is embedded in every other Gossip element.

Schedule:

This is the critical path element for Gossip. A prototype XML server exists for read-only XML access and is based on existing HTTP servers. The final version will require about 3 MM of effort to complete the server application and a similar level of effort to complete the associated APIs. The prototype can be finished to a level of detail suitable for other development in about .5 MM.

Staffing:

This element requires a detailed understanding of TCP/IP programming on the platforms we are deploying on (Win32, Mac, optionally Linux), a good understanding of concurrent programming techniques, thread safe development, and API development. Back-end data storage requirements call for experience in indexed file management and ODBC development.

Expert System

Description:

The Expert System element provides the intelligence behind the Gossip system. It operates on preconfigured rules that ship with the system as well as rules, assertions, and information provided over time by the user to tailor the behavior of the system to the user's work habits. Rules are stored in the XML Repository and locally compiled versions, along with operating data, are stored in the The expert system is the mechanism that triggers events within the Gossip system based on the analysis of the condition of data in the XML Repository and external events. As a result of this analysis, the expert system places commands and messages into the queues of other Gossip elements, triggering their operation as required.

Schedule:

While the Gossip system can operate in a basic capacity without the Expert System, this element needs to be completed early on. This element should take approximately 3 MM to complete. Design is complete for this element and proof of concept prototypes exist for this element, but the Gossip specific version has not been implemented.

Staffing:

This element requires an extensive knowledge of expert and production systems. This element manifests no user interface and has standard APIs available for accessing data elements, so this is strictly a high-end computer science effort. Data element design is also a requirement of this effort.

Scheduler

Description:

The Scheduler element is used to trigger events within the Gossip system on a regular basis. Other elements may request notifications, the user may schedule events or operations, and external events and messages may make entries into the scheduler's queues. The scheduler runs as a background task and performs methods specified in XML objects when their time triggers are activated. The scheduler is the primary mechanism for triggering agent-based retrievals of traditional network based data (e.g., HTML, e-mail, etc.) and implements the reminder portion of the Messaging UI. It has no user interface and its access to the file system and O/S is through standard APIs. Pending events are cached locally in the scheduler's event queue.

Schedule:

This application is a lower priority element and should take approximately 2 MM to complete. It does not exist at this time.

Staffing:

This element requires an understanding of multithreaded application development, basic database techniques, and an understanding of network programming.

Internet Agents

Description:

The HTTP, POP3, and SMTP agents provide an interface for the Gossip system to standard Internet-based protocols. The agents act in dual capacities, as clients for the Gossip system to access Internet resources external to the Gossip system and as servers to traditional Internet clients running on the local desktop system. This allows Gossip to participate in standard Web and e-mail dialogs with non-Gossip systems as well as allowing the user's traditional Internet client software like mail and Web clients to be used to access information in the Gossip system.

Schedule:

These three agents provide the primary means for communicating with the outside world. As such, Gossip needs this capability on-line at the same time as the Expert System element. Each agent is a fairly complete implementation of both the client and server versions of its specific protocol, though geared for a single user. So an estimate of 3 MM for each agent is anticipated. There are prototypes of the HTTP agent in existence now. POP3 and SMPT client agents exist in various states of completion, but no server components exist.

Staffing:

Staffing requirements are essentially the same as those for the XML Server. Familiarity with implementing HTTP, SMTP, and POP3 protocols is a big plus.

Configurator

Description:

The Configurator element is the primary user interface for inputting assertions, rules, and schedule information. This is the element that the user interacts with to train the system, load information into Gossip, and maintain the XML Repository. The Configurator maintains the consistency of XML elements and ensures that information provided by the user is stored in a consistent fashion.

Schedule:

This element does not currently exist. It is not a critical path item but it is a critical component of the finished user experience. In its absence, configuration data can be loaded into the system using bulk XML loads from the file system. It is estimated that this application will take approximately 5 MM to develop.

Staffing:

This element is user interface intensive and requires an extensive knowledge of platform specific UI development for both Win32 and Mac platforms. Natural language interpretation is a critical portion of this element. Interaction with the XML repository will be through standard APIs, so the primary skill set required is for UI design and language parsing and interpretation. An alternate implementation through standard Web browsers adds a requirement for HTML 3 authoring and design to this position.

Messaging UI

Description:

The Messaging UI element, better known as the "Bug", is the primary user interface to the Gossip system. This element is the small "postage stamp" window that floats in the corner of the user's screen, notifying them of incoming messages, events, and the results of searches and agent activities. The Bug presents several user interfaces on demand, including the instant messaging interface, notification interface, and a mechanism for triggering other UI applications or the user's Web browser.

Schedule:

This element is the primary user interface. It isn't critical to the development of other elements, but it is necessary in the early going to provide a way to observe Gossip internal operation from a user's perspective. This application will require approximately 6 MM to implement completely.

Staffing:

This element has similar staffing requirements to the Configurator element with the added need for familiarity in O/S specific interprocess communications.

Data and API Elements

Description:

A significant portion of Gossip's functionality comes from preconfigured rules and data which are used to operate data driven applications. The refinement of the Gossip data model, messaging architecture, and the standard APIs for manipulating these data and messages is important to the successful completion of all the other Gossip elements. The APIs also provide an abstraction layer over underlying Gossip and O/S services.

Schedule:

Implementation of the standard APIs proceeds in parallel with the development of the other critical Gossip elements. Data development cannot proceed until the Expert System element is operational. It is estimated that the API implementations will take 6 MM and the data configuration will take an additional 4 MM.

Staffing:

Data definition requires a strong background in entity based or object oriented data models as well as a strong understanding of Internet services. The API development requires detailed, specific knowledge about O/S specific services for file system operations, IPC, networking functions, thread management,

and dynamic code binding.

Documentation

Description:

Gossip presents the opportunity to be self documenting to a large degree. On-line, directed help will be critical to its success. When coupled with the expert system portion of Gossip, the on-line help should be very powerful. Accompanying text-based technical documentation is required as well.

Schedule:

This element is required later in the development cycle, once functionality is frozen and user interfaces are complete. It is not a critical path item. Estimated 3 MM to complete.

Staffing:

Experience in designing on-line help systems as well as a strong technical writing background is required. Tight integration with the development team is a must.

EXHIBIT C

```
/*
Copyright [redacted] BIAP Systems, Inc. All rights reserved.

BugApplication.java

Application shell for Gossip bug functions.

*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.io.*;


//subclassed BasicHTTPServer to provide logging in bug window

public class BugHTTPServer extends BasicHTTPServer {

      private BugAlert alert;

      public BugHTTPServer (BugAlert bugAlert, int pport, int backlog) {
            super (pport, backlog);
            alert = bugAlert;
      }

      public void LogMessage (String msg) {
            alert.Message (msg);
      }
}

//subclassed ODBObject to provide logging in console window

public class BugODB extends ODB {

      public BugODB () {
            super ();
      }

      public void LogMessage (String msg) {
//          System.out.println (msg);
      }

}


public class BugODBObject extends ODBObject {

      public BugODBObject (String vname, String vkind, boolean
hasChildren) {
            super (vname, vkind, hasChildren);
      }

      public void LogMessage (String msg) {
//          System.out.println (msg);
      }

}
```

```
/*
      Shell application that handles window management
*/

public class Main extends Frame implements WindowListener {

      //static String hostname;
      //static String host_address;
      //static int port = 80;

      private MenuBar mbar;
      private MenuItem mQuit, mTest;
      private MenuItemListener mItemListener;

      BugPanel bugp;                              // the panel that
handles drawing bug messages
      BugHTTPServer httpServer;           // the server thread that
handles incoming messages
      CGISimpleFile cgiSimpleFile;
      CGIRPCDispatcher rpcDispatch;
      CGIDeleteMessage deleteMessage;
      CGIIMForm imForm;
      CGIJavaSSI cgiJavaSSI;
      CGIODB cgiODB;
      SSIRegisterCommonHandlers common_handlers;
      SSIODBIterate odbiterate;
      CGIAgentBuilder agentBuilder;
      RPCUserInfo rpcUser;
      RPCIHaveIWant iHave;

      BugAlert alert;
      BugODB odb;
      Scheduler scheduler;
      ODBActionEngine actionEngine;
      UserAgent userAgent;

      ODBList imList, gList;  //list object containing the instant
messages received by the server

      public Main () {
            super ("Gossip");
            addWindowListener (this);
            this.setSize (160, 80);
System.out.println ("Gossip running...");
            mbar = new MenuBar ();
            Menu file = new Menu ("File", true);
            file.add (mTest = new MenuItem ("Test"));
            file.add (mQuit = new MenuItem ("Quit"));
            mbar.add (file);
            setMenuBar (mbar);

System.out.println ("  ...getting local host info");
            try {
                  Globals.hostname = InetAddress.getLocalHost
().getHostName ();
            }
            catch (Exception e) {
                  System.out.println (e.toString ());
                  Globals.hostname = "localhost";
            }
            //get the current IP address of this machine
```

```
            try {
                  Globals.ipAddress = InetAddress.getLocalHost
().getHostAddress ();
            }
            catch (Exception e) {
                  Globals.ipAddress = "127.0.0.1";
            }


System.out.println ("  ...initializing ODB");
            try {
                  odb = new BugODB ();
                  odb.init ();
                  odb.create ("gossip");
                  odb.open ("gossip");
                  Globals.gODB = odb;
            }
            catch (ODBException oe) {
                  System.out.println (oe.toString ());
            }

            //handle start-up configurations
System.out.println ("  ...initializing ODB data");
            gossipSetup ();

            bugp = new BugPanel (this, odb);
            add (bugp);

            alert = new BugAlert (bugp);

            //create all the conferences, queues, and buckets
            try {
                  imList = new ODBList (odb, "communities/instant");
            }
            catch (Exception e) {
                  imList = null;
            }

//////////make a test community (if it isn't already present)
            try {
                  new ODBCOI (odb, "communities/gossip_dev");
            }
            catch (Exception e) {
e.printStackTrace (System.out);
            }
//////////

System.out.println ("  ...configuring Gossip servers");
            //create a new HTTP server on port 80 with 10 threads
            httpServer = new BugHTTPServer (alert, Globals.port, 10);

            //register the CGI handlers
            httpServer.registerAction (bugp, "text/html", "bug"); //tell
the server who to call with messages
            httpServer.registerAction (cgiSimpleFile = new CGISimpleFile
(this), "text/html", "html");
            httpServer.registerAction (cgiSimpleFile, "text/html",
"htm");
            httpServer.registerAction (cgiSimpleFile, "image/gif",
"gif");
            httpServer.registerAction (cgiSimpleFile, "image/jpeg",
```

```
"jpg");
            httpServer.registerAction (cgiSimpleFile, "text/xml",
"xml");
            httpServer.registerAction (cgiJavaSSI = new CGIJavaSSI
(odb), "text/html", "ssi");
            httpServer.registerAction (deleteMessage = new
CGIDeleteMessage (odb), "text/html", "deletemessage");
            httpServer.registerAction (imForm = new CGIIMForm (odb,
imList), "text/html", "imform");
            httpServer.registerAction (odbiterate = new SSIODBIterate
(odb, cgiJavaSSI), "text/html", "browse_odb");
//          httpServer.registerAction (new CGIExecuteAction (odb,
cgiJavaSSI), "text/html", "executeAction");
            httpServer.registerAction (new CGIAgentBuilder (odb),
"text/html", "ab");

            //create the default ActionEngine
            actionEngine = new ODBActionEngine (odb, bugp, cgiJavaSSI);

            //register more CGI handlers
            httpServer.registerAction (new CGIExecuteAction (odb,
cgiJavaSSI, actionEngine), "text/html", "executeAction");
            httpServer.registerAction (cgiODB = new CGIODB (odb,
cgiJavaSSI), "text/html", "odb");
            common_handlers = new SSIRegisterCommonHandlers( cgiJavaSSI
);

            //register SSI handlers for the various CGIs
            cgiJavaSSI.registerHandler("publish", new SSIPublish (odb));
            cgiJavaSSI.registerHandler("ODBIterate", odbiterate);

System.out.println ("  ...constructing UI");
            pack ();
            this.show ();

            //stuff a sample message in the bug
            try {
                  bugp.addMessage (odb.getAttribute
("html/bugwelcome"));
            }
            catch (Exception e) {
                  bugp.addMessage ("SC 255 255 255\nFP 4 0 0 0 80 160 80
160 0\n" +
                                   "SC 192 192 192\nFP 4 10 0 10 80 30
80 30 0\n" +
                                   "SC 200 0 0\nFP 4 0 20 0 30 160 30
160 20\n" +
                                   "SC 0 0 0\nSF SansSerif 9 0\nDS 32
14 biap systems, inc. presents...\n" +
                                   "SC 255 0 0\nSF Serif 24 3\nDS 40 60
Gossip!\n");
            }

            //finish the menu stuff now that the ODB is created
            mItemListener = new MenuItemListener (odb);
            mTest.addActionListener (mItemListener);
            mQuit.addActionListener (mItemListener);

System.out.println ("  ...starting Server");
            //set up all the RPC stuff
            httpServer.registerAction (rpcDispatch = new
```

```
CGIRPCDispatcher (odb), "text/xml", "RPC2");
            rpcDispatch.registerMethod ("instantMessage", new
RPCInstantMsg (imList), true);
            rpcDispatch.registerMethod ("SetUserInfo", rpcUser = new
RPCUserInfo (), true);
            rpcDispatch.registerMethod ("ConfirmUserID", rpcUser, true);
            rpcDispatch.registerMethod ("IHaveIWant", iHave = new
RPCIHaveIWant (), true);
            rpcDispatch.registerMethod ("SetCOIData", iHave, true);

            //start receiving messages
            httpServer.start ();

System.out.println ("  ...starting Scheduler");
            //give the existing threads a chance to get initialized and
running
            try {Thread.sleep (100);Thread.yield();}catch (Exception
iee){}

            //start up the scheduler
            scheduler = new Scheduler (odb, actionEngine);
            scheduler.start ();

System.out.println ("  ...starting User Agent");
            userAgent = new UserAgent (odb);
            userAgent.start ();

System.out.println ("  ...start-up complete!");

      }


      public void gossipSetup () {
            String address;
            String name;
            String id, temp;

            try {
                  //update the user settings
                  odb.createAttribute ("settings/user/ipaddress",
"string", Globals.ipAddress);

                  //make sure there is a port number specified
                  try {
                        address = odb.getAttribute
("settings/user/port");
                        Globals.port = Integer.parseInt (address);
                        Globals.portStr = address;
                  }
                  catch (Exception ex) {
                        odb.createAttribute ("settings/user/port",
"int", "80");
                        Globals.port = 80;
                        Globals.portStr = "80";
                  }

                  //make sure there is a static IP flag
                  try {
                        temp = odb.getAttribute
("settings/user/staticIP");
                  }
```

```
                catch (Exception ex) {
                      odb.createAttribute ("settings/user/staticIP",
"string", "0");
                }

                //create the people entry for the user
                String localUserID = odb.getAttribute
("settings/user/userID");
                temp = "people/" + localUserID;
                try { odb.createObject (temp, "person"); } catch
(Exception x) {}
                try { odb.createAttribute (temp + "/real_name",
"string", odb.getAttribute ("settings/user/real_name")); } catch
(Exception x) {}
                try { odb.createAttribute (temp + "/ipaddress",
"string", Globals.ipAddress); } catch (Exception x) {}
                try { odb.createAttribute (temp + "/port", "int",
Globals.portStr); } catch (Exception x) {}
                try { odb.createAttribute (temp + "/userID", "string",
localUserID); } catch (Exception x) {}
                try { odb.createAttribute (temp + "/email_address",
"string", odb.getAttribute ("settings/user/email_address")); } catch
(Exception x) {}
                try { odb.createAttribute (temp + "/status", "string",
"local"); } catch (Exception x) {}
                try { odb.createAttribute (temp + "/date", "string",
DateUtils.dateToString (new Date ())); } catch (Exception x) {}
                try { odb.createAttribute (temp + "/staticIP",
"string", odb.getAttribute ("settings/user/staticIP")); } catch
(Exception x) {}

                //output the start-up document
                FileWriter f = new FileWriter (System.getProperty
("user.dir") + "/startup.htm");;
                PrintWriter out =  new PrintWriter (f);

                out.println ("<html><head><META HTTP-EQUIV=\"Refresh
\" CONTENT=\"1;URL=http://" + Globals.ipAddress + ":" + Globals.portStr
+ "/index.odb\"></head><body><a href=\"http://" + Globals.ipAddress +
"/index.odb\"><h1>Gossip!</h1></a></body></html>");

                out.close ();
            }
            catch (Exception e) {
                e.printStackTrace ();
            }
      }

      public void quit () {
            try {
                System.out.println ("Cleaning up...");
                scheduler.quit ();
                Thread.sleep (100);
                scheduler = null;

                httpServer.quit ();
                Thread.sleep (100);
                httpServer = null;

                bugp.quit ();
                Thread.sleep (100);
```

```
                bugp = null;

                odb.save ();
                odb.close ();
                odb.shutdown ();
            }
            catch (Exception e) {
                System.out.println (e.toString ());
            }

            dispose ();
            System.exit (0);
        }

        public void windowOpened (WindowEvent evt) {
        }

        public void windowClosed (WindowEvent evt) {
        }

        public void windowIconified (WindowEvent evt) {
        }

        public void windowDeiconified (WindowEvent evt) {
        }

        public void windowActivated (WindowEvent evt) {
        }

        public void windowDeactivated (WindowEvent evt) {
        }

        public void windowClosing (WindowEvent evt) {
            quit ();
        }

        public static void main(String args[]) {
            new Main ();
        }

        class MenuItemListener implements ActionListener {
            private ODB odb;

            public MenuItemListener (ODB vodb) {
                super ();
                odb = vodb;
            }

            public void actionPerformed (ActionEvent event) {
                MenuItem i = (MenuItem) event.getSource ();
                if (i == mQuit) {
                    quit ();
                }
                else if (i == mTest) {
                    // More test code for Instant Message stuff
                    new Gossiper (odb, "gossip_dev", "7575393310");
                }
            }
        }

}
```

```
/*
      Copyright [redacted] BIAP Systems, Inc. All rights reserved.

      HTMLTreeParser.java

      Written by Louis Slothouber on [redacted]

*/

import java.io.Reader;
import java.io.StringReader;
import java.util.Vector;

public class HTML_Element {
      final static int CONTAINER    = 0;
      final static int ATTRIBUTE    = 1;
      final static int CONTENT      = 2;

      HTML_Element      parent;
      int               type;
      boolean           displayend;
      String                  name;
      String                  data;
      Vector                  children;
      Vector                  attributes;

      //Constructor (empty) --can't be called
      private HTML_Element () {}

      //Constructor (container)
      HTML_Element ( HTML_Element parent, String name, boolean
hasChildren, boolean hasAttributes ) {
            init(parent, CONTAINER, name, null, hasChildren,
hasAttributes);
      }

      //Constructor (content)
      HTML_Element ( HTML_Element parent, String data) {
            init(parent, CONTENT, null, data, false, false);
      }

      //Constructor (attribute)
      HTML_Element ( HTML_Element parent, String name, String data ) {
            init(parent, ATTRIBUTE, name, data, false, false);
      }

      //Initialize the new element based on the data provided
      private void init ( HTML_Element parent, int type, String name,
String data, boolean hasChildren, boolean hasAttributes) {
            this.type = type;
            this.parent = parent;
            this.name = name;
            this.data = data;
            this.displayend = false;
            if (hasChildren)
                  children = new Vector(2, 2);
            else
                  children = null;
            if (hasAttributes)
                  attributes = new Vector(3, 2);
            else
```

```
            attributes = null;
      }

      //Add a child element to the current object
      public void addChild ( HTML_Element e ) {
            if (children == null)
                  children = new Vector(1,2);
            if (type == CONTAINER)
                  children.addElement( e );
      }

      //Add a child element to the current object
      public void addAttribute ( HTML_Element e ) {
            if (attributes == null)
                  attributes = new Vector(2, 2);
            if (type == CONTAINER)
                  attributes.addElement( e );
      }

      //Add a child attribute element to the current object
      public void addAttribute ( String name, String data ) {
            addAttribute( new HTML_Element(this, name, data) );
      }

      //Add a child content element to the current object
      public void addContent ( String data ) {
            addChild( new HTML_Element(this, null, data) );
      }

      //Delete the current element and all it's children (recursively)
      public void dispose () {
            boolean found = false;
            int   i;
            Vector      tempv;
            int         v_ct;

            //Clear this element
            name = null;
            data = null;
            if (children != null)
                  children.removeAllElements();
            if (attributes != null)
                  attributes.removeAllElements();
            children = null;
            attributes = null;

            //Remove child from parent's children/argument list
            if (parent != null)
                  if ((type == ATTRIBUTE) && (parent.attributes !=
null))
                        parent.attributes.removeElementAt
(parent.attributes.indexOf(this));
                  else if (parent.children != null)
                        parent.children.removeElementAt
(parent.children.indexOf(this));
      }

      public String getElementData () {
            return data;
      }
```

```
public int getElementType () {
      return type;
}

public String getElementName () {
      return name;
}

public HTML_Element getParent () {
      return parent;
}

public HTML_Element getChild ( int i ) {
      try {
            if (children == null)
                  children = new Vector(1,2);
            if (type == CONTAINER)
                  return (HTML_Element) children.elementAt(i);
      } catch (Exception e) {
      }
      return null;
}

public int getChildCount () {
      if (children == null)
            children = new Vector(1,2);
      if (type == CONTAINER)
            return children.size();
      else
            return -1;
}

public int getMyIndex () {
      int result = -1;
      int i;

      if (parent != null)
            for (i=0; i<parent.getChildCount(); i++)
                  if (parent.getChild(i) == this) {
                        result = i;
                        break;
                  }

      return result;
}

public HTML_Element getNextSibling () {
      HTML_Element result = null;
      int index;

      if (parent != null) {
            index = getMyIndex() + 1;
            if (index < parent.getChildCount())
                  result = parent.getChild(index);
      }

      return result;
}

public HTML_Element findNthTag ( String tagname, int count ) {
      HTML_Element temp = this;
```

```
            HTML_Element next;

            while ((temp != null) && (count > 0)) {
                  next = temp.getChild(0);
                  if (next == null)
                        next = temp.getNextSibling();
                  while ((temp != null) && (next == null)) {
                        temp = temp.getParent();
                        if (temp != null)
                              next = temp.getNextSibling();
                  }
                  temp = next;
                  if ((temp != null) && (temp.getElementType() ==
CONTAINER))
                        if (temp.getElementName().equalsIgnoreCase
(tagname))
                              count--;
            }

            return temp;
      }

      public void printTree (int depth, int maxdepth) {
            HTML_Element temp;
            int j;

            if ((this != null) && (depth < maxdepth)) {

                  switch (getElementType()) {
                        case CONTAINER:
                              for (j=0; j<depth; j++)
                                    System.out.print("...");
                              System.out.println(depth + " ( " +
getElementName());
                              for (j=0; j<getChildCount(); j++)
                                    getChild(j).printTree(depth+1,
maxdepth);
                              for (j=0; j<depth; j++)
                                    System.out.print("...");
                              System.out.println(depth + " )");
                              break;
                        case CONTENT:
                              if (getElementData().trim().length() > 0)
{
                                    for (j=0; j<depth; j++)
                                          System.out.print("...");
                                    System.out.println(depth + " \"" +
getElementData().trim() + "\"");
                              }
                              break;
                        default:
                              for (j=0; j<depth; j++)
                                    System.out.print("...");
                              System.out.println(depth + "OTHER");
                              break;
                  }
            }
      }

      public HTML_Element getAttribute ( String name ) {
            int i;
```

```
            if (type == CONTAINER) {
                  if (attributes == null)
                        attributes = new Vector(2, 2);
                  for (i=0; i<attributes.size(); i++)
                        if (((HTML_Element) attributes.elementAt
(i)).name.equalsIgnoreCase(name))
                              return (HTML_Element) attributes.elementAt
(i);
            }

            return null;
      }

      public HTML_Element getAttribute ( int i ) {
            try {
                  if (type == CONTAINER) {
                        if (attributes == null)
                              attributes = new Vector(2, 2);
                        return (HTML_Element) attributes.elementAt(i);
                  }
            } catch (Exception e) {
            }
            return null;
      }

      public String getAttributeValue ( String name ) {
            HTML_Element temp = getAttribute(name);

            if (temp != null)
                  return temp.data;
            else
                  return null;
      }

      public String getAttributeValue ( int i ) {
            HTML_Element temp = getAttribute(i);

            if (temp != null)
                  return temp.data;
            else
                  return null;
      }

      public int getAttributeCount () {
            if (type == CONTAINER) {
                  if (attributes == null)
                        attributes = new Vector(2, 2);
                  return attributes.size();
            } else
                  return -1;
      }

      public String getAllContents () {
            int i;
            String result = new String();

            if (type == CONTAINER) {
                  if (children == null)
                        children = new Vector(1,2);
                  for (i=0; i<children.size(); i++)
```

```
                        result = result + children.elementAt(i).toString
();
            }

            return result;
      }

      public String getStartTag () {
            String result = null;
            int i;

            if (type == CONTAINER) {
                  result = "<" + getElementName();
                  if (attributes != null)
                        for (i=0; i<attributes.size(); i++)
                              result = result + " " +
attributes.elementAt(i).toString();
                  result = result + ">";
            }

            return result;
      }

      public String getEndTag () {
            String result = null;

            if (type == CONTAINER) {
                  if (displayend)
                        result = "</" + name + ">";
                  else
                        result = "";
            }

            return result;
      }

      public String toString () {
            String result = "";
            int i;

            if (this != null)
                  switch (this.getElementType()) {
                        case CONTAINER:
                              result = getStartTag() + getAllContents()
+ getEndTag();
                              break;
                        case ATTRIBUTE:
                              result = name + "=\"" + data + "\"";
                              break;
                        case CONTENT:
                              result = data;
                              break;
                  }

            return result;
      }

      public void setElementData ( String data ) {
            this.data = data;
      }
```

```
        public void setElementType ( int type ) {
              this.type = type;
        }

        public void setElementName ( String name ) {
              this.name = name;
        }

        public void setChild ( int i, HTML_Element child ) {
              if ((type == CONTAINER) && (child.type != ATTRIBUTE)) {
                    if (children == null)
                          children = new Vector(1,2);
                    children.setElementAt(child, i);
              } else
                    System.out.println("Error: setChild");
        }

        public void setAttribute( int i, HTML_Element attribute ) {
              if ((type == CONTAINER) && (attribute.type == ATTRIBUTE)) {
                    if (attributes == null)
                          attributes = new Vector(2, 2);
                    attributes.setElementAt(attribute, i);
              } else
                    System.out.println("Error: setAttribute");
        }

}

public class HTMLTreeParser extends HTMLParser {

        private HTML_Element current_element = null;
        private HTML_Element root_element = null;
        private String has_attribute_tags = "body,table,a,img,form,input";
        private String no_children_tags = "img,hr,li";
        private boolean   getCommentFlag = false;
        private Reader reader = null;

        public HTMLTreeParser ( String s ) {
              reader = new StringReader(s);

              Parse();
        }

        public HTMLTreeParser ( String s, boolean getCommentFlag) {
              reader = new StringReader(s);
              this.getCommentFlag = getCommentFlag;

              Parse();
        }

        public HTMLTreeParser ( Reader r, boolean getCommentFlag ) {
              reader = r;
              this.getCommentFlag = getCommentFlag;

              Parse();
        }

        public HTMLTreeParser ( Reader r ) {
              reader = r;

              Parse();
```

```
	}

	public boolean getComments() {
		return getCommentFlag;
	}

	public HTML_Element getRootElement () {
		return root_element;
	}

	private void Parse () {
		try {
			current_element = root_element = new HTML_Element
(null, "root", false, false);
			parseDocument(reader);
		} catch (Exception e) {
System.out.println("E: " + e.toString());
			root_element = null;
		}
	}

	protected boolean StartElement ( hpStartElementEvent event )
throws Exception {
		String name = event.getName().toLowerCase();
		boolean hasAttributes = has_attribute_tags.indexOf(name) >=
0;
		boolean hasChildren = no_children_tags.indexOf(name) < 0;
		HTML_Element parent_element = current_element;
		int i;

		current_element = new HTML_Element(parent_element, name,
hasChildren, hasAttributes);
		for (i=0; i<event.getAttributeCount(); i++)
			current_element.addAttribute( new HTML_Element(
current_element, event.getAttributeName(i), event.getAttributeValue(i) )
);
		parent_element.addChild(current_element);

		return false;
	}

	protected boolean EndElement ( hpEndElementEvent event ) throws
Exception {
		if (current_element != null) {
			current_element.displayend = event.getDisplay();
			current_element = current_element.getParent();
		}
		return false;
	}

	protected boolean CharacterData ( hpCharacterDataEvent event )
throws Exception {
		current_element.addChild( new HTML_Element(current_element,
event.copyString()) );
		return false;
	}

	protected boolean Comment ( hpCommentEvent event ) throws
Exception {
		if (getCommentFlag)
			current_element.addChild( new HTML_Element
```

```
(current_element, event.copyString()) );
            return false;
      }

      protected boolean ProcessingInstruction (
hpProcessingInstructionEvent event ) throws Exception {
            return false;
      }

      protected void EndDocument() throws Exception {
      }

      protected void LogMessage(String s) {
//          System.out.println( s );
      }
}
```

```
/*
      Copyright [redacted] BIAP Systems, Inc. All rights reserved.

      HTMLParser.java

      Written by Louis Slothouber on [redacted]

*/

import java.io.Reader;
import java.util.Hashtable;
import java.util.Enumeration;
import java.util.Stack;
import java.util.Vector;

public class hpElementEvent {
      String buffer;

      hpElementEvent (String buffer) {
            this.buffer = buffer;
      }

      public String getElement() {
            return new String(this.buffer);
      }

      public int copyElementChars (char[] cbuf, int off) {
            int copylen = Math.min((cbuf.length - off), buffer.length
());

            buffer.getChars(0, copylen, cbuf, off);

            return copylen;
      }
}

public class hpStartElementEvent extends hpElementEvent {
      Hashtable   args;
      int               argct;

      hpStartElementEvent (String buffer, Hashtable args, int argct) {
            super(buffer);
            this.args = args;
            this.argct = argct;
      }

      public int getAttributeCount() {
            Enumeration arglist = null;
            int count = 0;

            arglist = args.elements();
            while (arglist.hasMoreElements()) {
                  count++;
                  arglist.nextElement();
            }

            return count;
      }

      public String getAttributeName(int i) {
            Enumeration arglist = null;
```

```
            String              result = null;
            int                 count = 0;

            arglist = args.keys();
            while ((count++ <= i) && arglist.hasMoreElements())
                  result = new String((String) arglist.nextElement());

            return result;
      }

      public String getAttributeValue(int i) {
            Enumeration arglist = null;
            String              result = null;
            int                 count = 0;

            arglist = args.elements();
            while ((count++ <= i) && arglist.hasMoreElements())
                  result = new String((String) arglist.nextElement());

            return result;
      }

      public String getAttributeValue(String name) {
            return (new String((String) args.get(name.toLowerCase())));
      }

      public String getName() {
            return getElement();
      }
}

public class hpEndElementEvent extends hpElementEvent {
      boolean display;

      hpEndElementEvent (String buffer) {
            super(buffer);
            display = true;
      }

      public String getName() {
            return getElement();
      }

      public void setDisplay (boolean display) {
            this.display = display;
      }

      public boolean getDisplay () {
            return display;
      }
}

public class hpCharacterDataEvent extends hpElementEvent {

      hpCharacterDataEvent (String buffer) {
            super(buffer);
      }

      public int getLength () {
            return buffer.length();
      }
```

```
        public int copyChars (char[] cbuf, int off) {
              return copyElementChars(cbuf, off);
        }

        public String copyString() {
              char[] cbuf = new char[buffer.length()];
              copyElementChars(cbuf, 0);
              return new String( cbuf );
        }
}

public class hpCommentEvent extends hpElementEvent {

        hpCommentEvent (String buffer) {
              super(buffer);
        }

        public int getLength () {
              return buffer.length();
        }

        public int copyChars (char[] cbuf, int off) {
              return copyElementChars(cbuf, off);
        }

        public String copyString() {
              char[] cbuf = new char[buffer.length()];
              copyElementChars(cbuf, 0);
              return new String( cbuf );
        }
}

public class hpProcessingInstructionEvent extends hpElementEvent {

        hpProcessingInstructionEvent (String buffer) {
              super(buffer);
        }

        public int getLength () {
              return buffer.length();
        }

        public int copyChars (char[] cbuf, int off) {
              return copyElementChars(cbuf, off);
        }

        public String copyString() {
              char[] cbuf = new char[buffer.length()];
              copyElementChars(cbuf, 0);
              return new String( cbuf );
        }
}

public class TagStack {
        Vector tags = new Vector();
        Vector matched = new Vector();
        int    top  = -1;

        TagStack () {
        }
```

```
	public void push (String tag, boolean matched) {
		tags.addElement(tag);
		this.matched.addElement(new Boolean(matched));
		top++;
	}

	public boolean topmatched () {
		if (top >= 0)
			return ((Boolean) matched.elementAt(top)).booleanValue
();
		else
			return false;
	}

	public String peek () {
		return toptag();
	}

	public String toptag () {
		if (top >= 0)
			return (String) tags.elementAt(top);
		else
			return null;
	}

	public boolean empty () {
		return (top < 0);
	}

	public void match (int i) {
		if (i >= 0)
			matched.setElementAt(new Boolean(true), i);
	}

	public int search (String tag) {
		int i;

		for (i=top; i>=0; i--) {
			if (((String) tags.elementAt(i)).equalsIgnoreCase(tag)
&& !((Boolean) matched.elementAt(i)).booleanValue())
				break;
		}

		return i;
	}

	public void pop () {
		if (!empty()) {
			tags.removeElementAt(top);
			matched.removeElementAt(top);
			top--;
		}
	}
}
public abstract class HTMLParser {
	private TagStack tagstack;
	private boolean   abortparse = false;
	private static String singletags =
"~p~li~hr~br~img~meta~option~area~input~";
```

```
      HTMLParser() {
            abortparse = false;
      }

      protected abstract boolean StartElement ( hpStartElementEvent
event ) throws Exception;
      protected abstract boolean EndElement ( hpEndElementEvent event )
throws Exception;
      protected abstract boolean CharacterData ( hpCharacterDataEvent
event ) throws Exception;
      protected abstract boolean Comment ( hpCommentEvent event ) throws
Exception;
      protected abstract boolean ProcessingInstruction (
hpProcessingInstructionEvent event ) throws Exception;
      protected abstract void EndDocument() throws Exception;
      protected abstract void LogMessage(String s);

      public void parseDocument ( Reader reader ) throws Exception {
            char[]                  cbuff = new char[2];
            char              ch;
            int                     state = 0;
            String                  current_tag = null;
            StringBuffer      buffer = new StringBuffer();
            boolean                 ignore = false;
            int                     wsstate = 0;
            int                     wsstate2 = 0;
            char              wstest = 'a';
            char              wstest2 = 'a';
            String                  prevtag = null;
            String                  tagname = null;
            String                  argname = null;
            Hashtable         args = null;
            int                     argct = 0;

            abortparse = false;
            tagstack = new TagStack();
            while (!abortparse && (reader.read(cbuff, 0, 1) == 1)) {
                  ch = cbuff[0];
//System.out.println( "State: " + Integer.toString(state) + "  ch = " +
ch);
                  switch (state) {
                        case 0:     //Initial state, between tags
//                            if (ch == '\"')
//                                  state = 1;
//                            else
                            if (ch == '<') {
                                  if (buffer.length() > 0)
                                        abortparse = CharacterData (
new hpCharacterDataEvent( new String(buffer) ) );
                                  buffer = new StringBuffer();
                                  state = 2;
                            }
                            //else, just gather content in 'buffer'.
                            break;
//                      case 1:     //Inside quote - disable if there
might be mismatched quotes
//                            if ((ch == '\"') && (buffer.charAt
(buffer.length()-1) != '\\'))
//                                  state = 0;
//                            break;
                        case 2: // after <
```

```
                                prevtag = tagname;
                                if (ch == '!')
                                      state = 3;
                                else if (ch == '/')
                                      state = 9;
                                else if (Character.isLetter(ch))
                                      state = 11;
                                else if (Character.isWhitespace(ch)) {
                                      ignore = true;
                                      wsstate = 11;
                                      wstest = '1';
                                      wstest2 = '~';
                                      wsstate2 = 11;
                                      state = 100;
                                } else                              //error:
just treat it as normal text, not markup
                                      state = 0;
                                break;

                          // ------------------------------ Comment
------------------------------
                          case 3: //after !
                                if (ch == '-')
                                      state = 4;
                                else
                                      state = 8;
                                break;
                          case 4: //after -
                                if (ch == '-')
                                      state = 5;
                                else
                                      state = 0;
                                break;
                          case 5: //Comment
                                if (ch == '-')
                                      state = 6;
                                break;
                          case 6: //Comment after -
                                if (ch == '-')
                                      state = 7;
                                else
                                      state = 5;
                                break;
                          case 7: //Comment after --
                                if (ch == '>') {
                                      buffer.append(ch);            //add
> to end of the comment
                                      ignore = true;
                  //don't add > to beginning of next buffer
                                      abortparse = Comment ( new
hpCommentEvent( new String(buffer) ) );
                                      buffer = new StringBuffer();
                                      state = 0;
                                } else if (ch != '-')
                                      state = 5;
                                break;
                          case 8: //<! directive
                                if (ch == '>') {
                                      buffer.append(ch);            //add
> to end of the comment
                                      ignore = true;
```

```
            //don't add > to beginning of next buffer
                                    abortparse = ProcessingInstruction
( new hpProcessingInstructionEvent( new String(buffer) ) );
                                    buffer = new StringBuffer();
                                    state = 0;
                              }
                              break;

                        // ---------------------------- End Tag
----------------------------
                        case 9: //After </
                              if (Character.isWhitespace(ch)) {
                                    wstest = 'l';
                                    wstest2 = '~';
                                    wsstate = wsstate2 = 10;
                                    state = 100;
                                    ignore = true;
                              } else if (Character.isLetter(ch))
                                    state = 10;
                              else                        //error: just
treat it as normal text, not markup
                                    state = 0;
                              break;
                        case 10: //Close Tag Name
                              if (Character.isWhitespace(ch)) {
                                    wstest = '>';
                                    wstest2 = '~';
                                    wsstate = wsstate2 = 0;
                                    state = 100;
                                    ignore = true;
                              } else if (ch == '>') {
                                    tagname = (new String
(buffer)).substring(2, buffer.length()).toLowerCase();
                                    EndATag( tagname );
                                    buffer = new StringBuffer();
                                    ignore = true;
                                    state = 0;
                              }  //Otherwise, just bundle characters
into tag name ('buffer')
                              break;

                        // ---------------------------- Start Tag
---------------------------

                        case 11: //Start Tag first non-space letter
                              argct = 0;
                              args = new Hashtable();
                              tagname = null;
                              if (Character.isWhitespace(ch)) {
                                    tagname = (new String
(buffer)).substring(1, buffer.length()).toLowerCase();
                                    buffer = new StringBuffer();
                                    wstest = 'l';
                                    wstest2 = '>';
                                    wsstate = 13;
                                    wsstate2 = 0;
                                    state = 100;
                                    ignore = true;
                              } else if (ch == '>') {
                                    tagname = (new String
(buffer)).substring(1, buffer.length()).toLowerCase();
```

```
                                    buffer = new StringBuffer();
                                    StartATag( tagname, args, argct );
                                    ignore = true;
                                    state = 0;
                               } else
                                    state = 12;
                               break;
                          case 12: //Get rest of Start Tag from second
character
                               if (Character.isWhitespace(ch)) {
                                    tagname = (new String
(buffer)).substring(1, buffer.length()).toLowerCase();
                                    buffer = new StringBuffer();
                                    wstest = '1';
                                    wstest2 = '>';
                                    wsstate = 13;
                                    wsstate2 = 0;
                                    state = 100;
                                    ignore = true;
                               } else if (ch == '>') {
                                    tagname = (new String
(buffer)).substring(1, buffer.length()).toLowerCase();
                                    buffer = new StringBuffer();
                                    StartATag( tagname, args, argct );
                                    ignore = true;
                                    state = 0;
                               }
                               // else gather the rest of the start tag
name
                               break;
                          case 13: //First argument name from first non-
space character
                               if (Character.isWhitespace(ch)) {
                                    argname = new String(buffer);
                                    buffer = new StringBuffer();
                                    argct++;
                                    ignore = true;
                                    state = 135;
                               } else if (ch == '=') {

                                    argname = new String(buffer);
                                    buffer = new StringBuffer();
                                    argct++;
                                    ignore = true;
                                    state = 14;
                               } else if (ch == '>') {
                                    argname = new String(buffer);
                                    buffer = new StringBuffer();
                                    argct++;
                                    args.put(argname, new String
(buffer));
                                    StartATag( tagname, args, argct );
                                    ignore = true;
                                    state = 0;
                               } //else, gather the rest of the argument
name
                               break;
                          case 135: //second non-blank after argname
                               if (Character.isWhitespace(ch))
        {      //continue to ignore whitespace
                                    ignore = true;
```

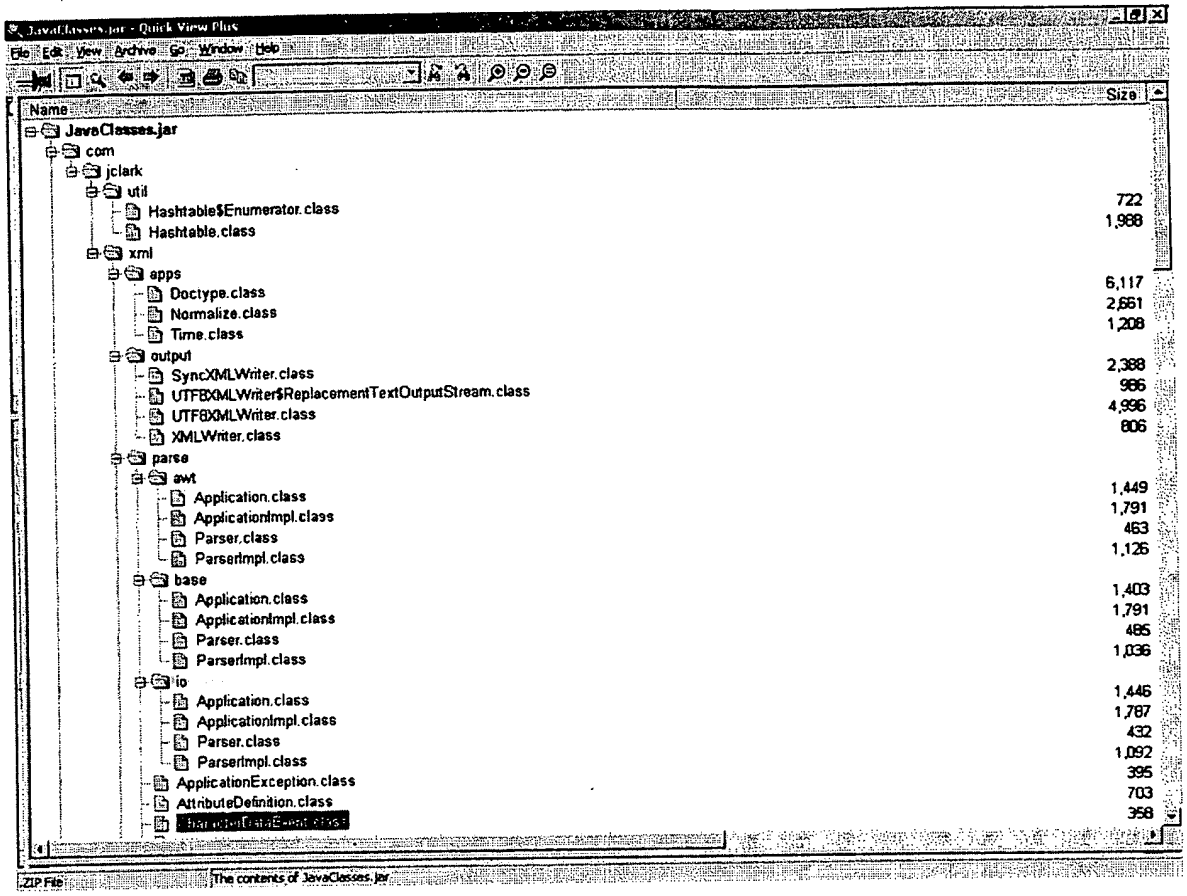
```
                        } else if (ch == '=') {
                              argname = new String(buffer);
                              buffer = new StringBuffer();
                              argct++;
                              ignore = true;
                              state = 14;
                        } else if (ch == '>') {
                              argname = new String(buffer);
                              buffer = new StringBuffer();
                              argct++;
                              args.put(argname, new String
(buffer));
                              StartATag( tagname, args, argct );
                              ignore = true;
                              state = 0;
                        } else {
                              buffer = new StringBuffer();
                              argct++;
                              args.put(argname, new String
(buffer));
                              state = 13;
                        }
                        break;
                  case 14: //First character after = (may be
blank)
                        if (Character.isWhitespace(ch)) {
                              wstest = 'a';
                              wstest2 = '>';
                              wsstate = 15;
                              wsstate2 = 0;
                        } else if (ch == '>') {
                              args.put(argname, new String
(buffer));
                              buffer = new StringBuffer();
                              StartATag( tagname, args, argct );
                              ignore = true;
                              state = 0;
                        } else
                              state = 15;
                        break;
                  case 15: //Get the rest of the argument value
(check quote status)
                        if (buffer.charAt(buffer.length()-1) ==
'\"') {
                              buffer.setLength(buffer.length()-1);
                              if ((ch == '\"') && ((buffer.length
() == 0) || (buffer.charAt(buffer.length()-1) != '\\'))) {
                                    ignore = true;
                                    state = 19;
                              } else // else, gather up double-
quoted string
                                    state = 17;
                        } else if (buffer.charAt(buffer.length()-
1) == '\'') {
                              buffer.setLength(buffer.length()-1);
                              if ((ch == '\'') && ((buffer.length
() == 0) || (buffer.charAt(buffer.length()-1) != '\\'))) {
                                    ignore = true;
                                    state = 19;
                              } else // else, gather up single-
quoted string
```

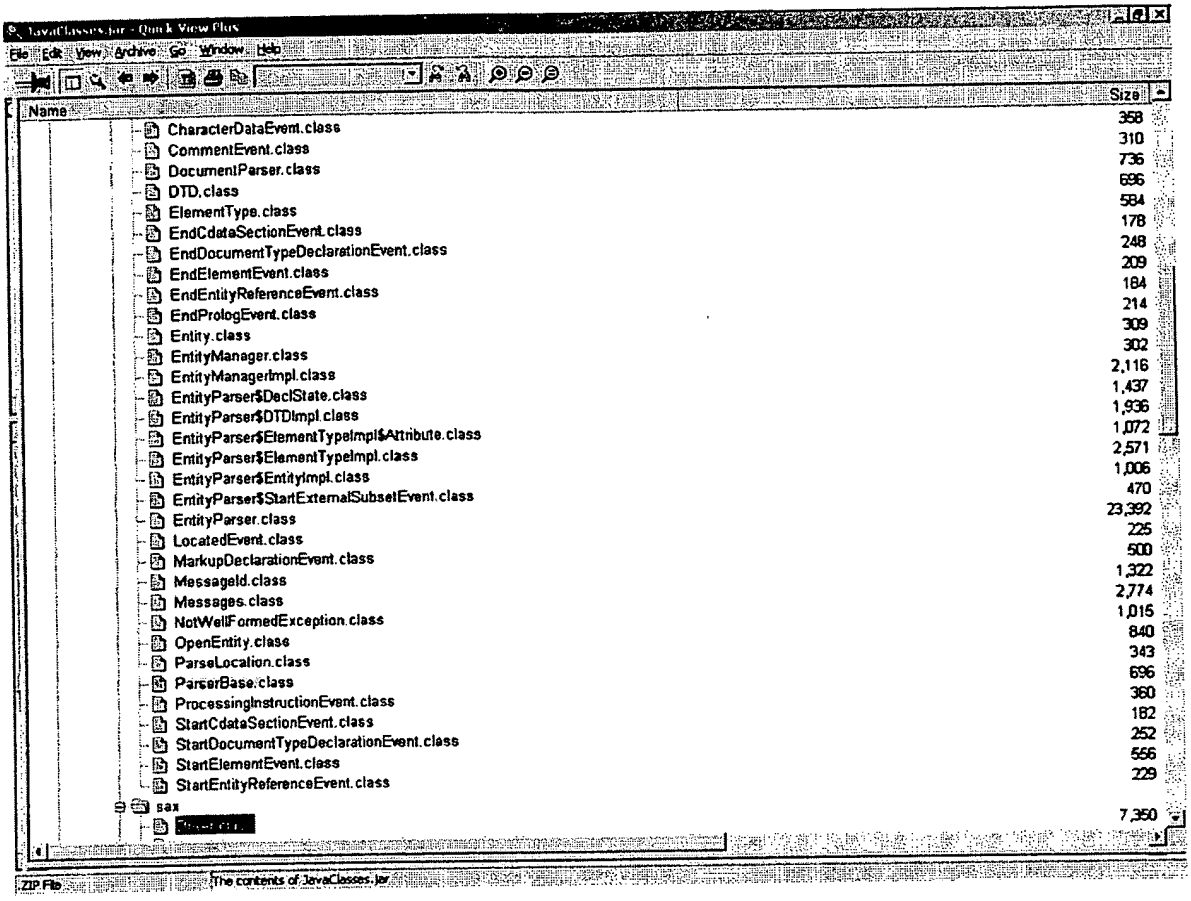
```
                                    state = 18;
                                } else if (Character.isWhitespace(ch)) {
                                    args.put(argname, new String
(buffer));
                                    buffer = new StringBuffer();
                                    ignore = true;
                                    wstest = 'l';
                                    wsstate = 13;
                                    wstest2 = '>';
                                    wsstate2 = 0;
                                    state = 100;
                                } else if (ch == '>') {
                                    args.put(argname, new String
(buffer));
                                    buffer = new StringBuffer();
                                    StartATag( tagname, args, argct );
                                    ignore = true;
                                    state = 0;
                                } else
                                    state = 16;
                                break;
                            case 16: //Get the rest of the non-quoted
argument value
                                if (Character.isWhitespace(ch)) {
                                    args.put(argname, new String
(buffer));
                                    buffer = new StringBuffer();
                                    ignore = true;
                                    wstest = 'l';
                                    wsstate = 13;
                                    wstest2 = '>';
                                    wsstate2 = 0;
                                    state = 100;
                                } else if (ch == '>') {
                                    args.put(argname, new String
(buffer));
                                    buffer = new StringBuffer();
                                    StartATag( tagname, args, argct );
                                    ignore = true;
                                    state = 0;
                                } //else, gather up rest of non-quoted
string
                                break;
                            case 17: //Get the rest of the double-quoted
argument value
                                if ((ch == '\"') && ((buffer.length() ==
0) || (buffer.charAt(buffer.length()-1) != '\\'))) {
                                    ignore = true;
                                    state = 19;
                                } // else, gather up double-quoted string
                                break;
                            case 18: //Get the rest of the single-quoted
argument value
                                if ((ch == '\'') && ((buffer.length() ==
0) || (buffer.charAt(buffer.length()-1) != '\\'))) {
                                    ignore = true;
                                    state = 19;
                                } // else, gather up single-quoted string
                                break;
                            case 19: //Check for another argument, or end of
the tag
```

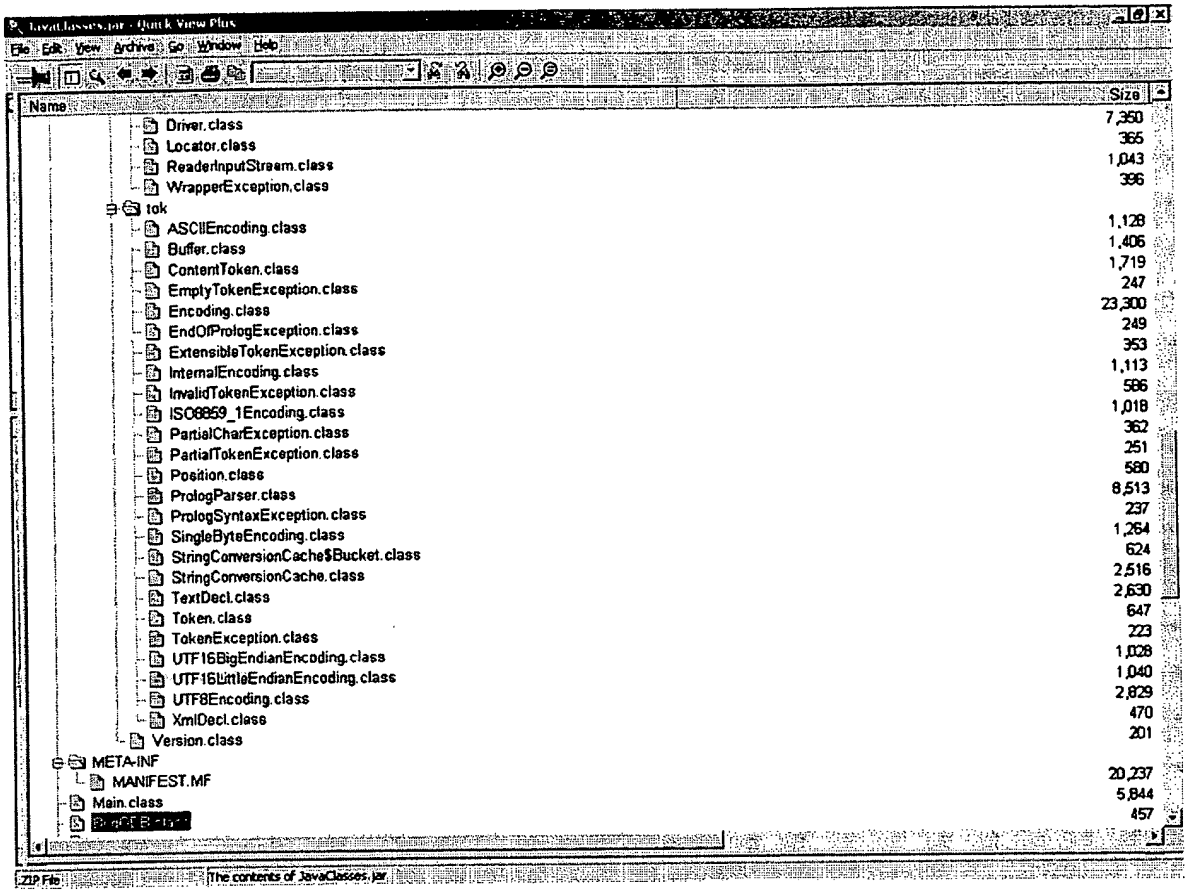
```
                                if (Character.isWhitespace(ch)) {
                                      args.put(argname, new String
(buffer));
                                      buffer = new StringBuffer();
                                      ignore = true;
                                      wstest = 'l';
                                      wsstate = 13;
                                      wstest2 = '>';
                                      wsstate2 = 0;
                                      state = 100;
                                } else if (ch == '>') {
                                      args.put(argname, new String
(buffer));
                                      buffer = new StringBuffer();
                                      StartATag( tagname, args, argct );
                                      ignore = true;
                                      state = 0;
                                } else {
                                      args.put(argname, new String
(buffer));
                                      buffer = new StringBuffer();
                                      state = 13;
                                }
                                break;
                          // ------------------------------ WhiteSpace
------------------------------
                          case 100: //WhiteSpace
                                if (Character.isWhitespace(ch))
                    //continue to ignore whitespace
                                      ignore = true;
                                if (((wstest2 == '~') || (ch != wstest2))
&&          //see if wstest applies
                                      ((wstest == 'l') &&
Character.isLetterOrDigit(ch)) ||
                                      (wstest == 'a') ||
                                      (ch == wstest)) {
                                      state = wsstate;
                                      if ((wstest != 'a') && (wstest !=
'l'))
                                            ignore = true;
                                } else if ((wstest2 != '~') &&
                                            ((wstest2 == 'l') &&
Character.isLetterOrDigit(ch)) ||
                                            (wstest2 == 'a') ||
                                            (ch == wstest2)) {
                                      state = wsstate2;
                                      if ((wstest2 != 'a') && (wstest2 !=
'l'))
                                            ignore = true;
                                }
                                if ((wstest2 == '>') && (ch == '>')) {
                          //Handle end of start tag
                                      if (argct == 0)
                                            ; //tagname = (new String
(buffer)).substring(1, buffer.length()).toLowerCase();
                                      else
                                            args.put(argname, new String
(buffer));
                                      buffer = new StringBuffer();
                                      StartATag( tagname, args, argct );
                                }
```

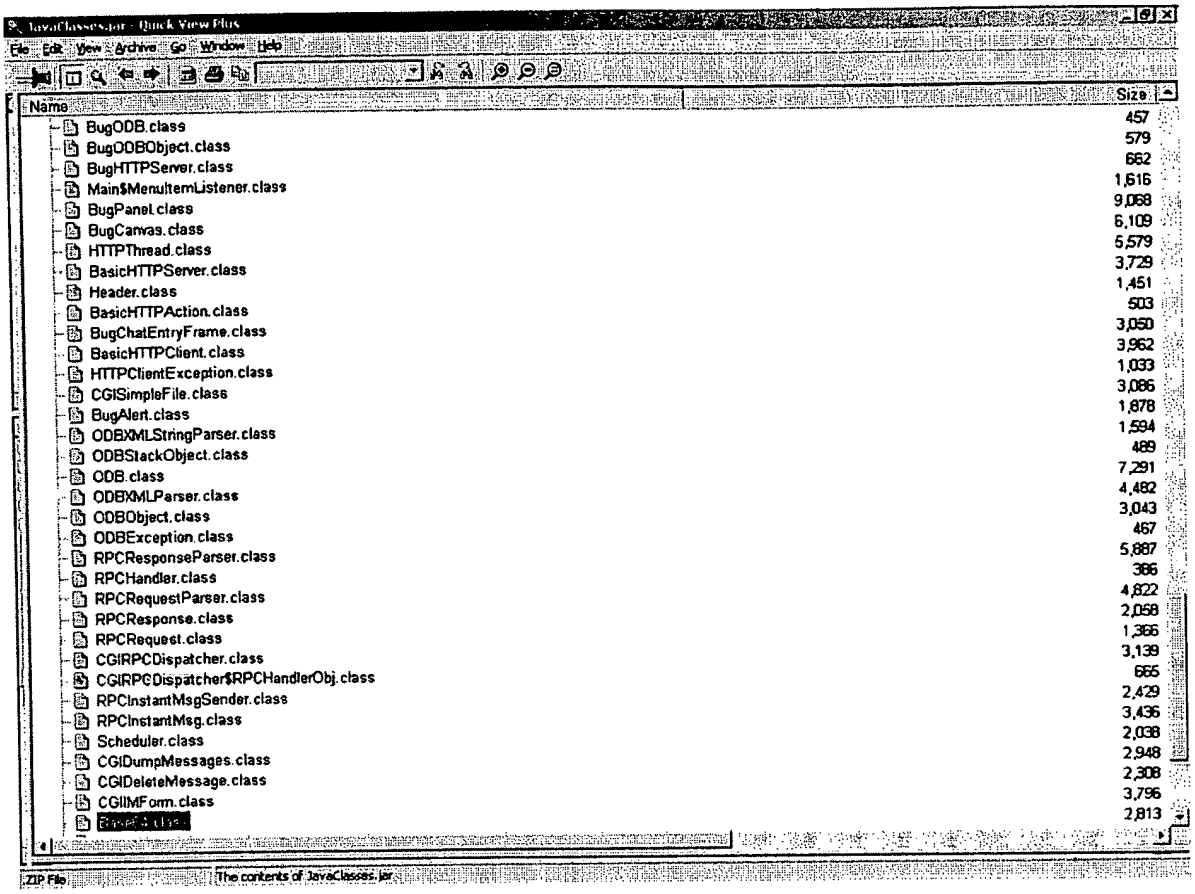
```
                              break;

                        default: //error
                              LogMessage("ERROR: Missing parser state "
+ Integer.toString(state));
                              state = 0;
                              break;
                  } //switch

                  //Copy ch to buffer
                  if (!ignore) {
                        buffer.append(ch);
                  } else {
                        ignore = false;
                  }
            } //while
            EndDocument();
      }

      private void StartATag( String tag, Hashtable args, int argct ) {
            String top = null;

            try {
                  //Generate start tag
                  abortparse = StartElement ( new hpStartElementEvent(
tag, args, argct ) );

                  //Handle trivial closing of <P>, <LI>, <HR>, <BR>,
<IMG>, <INPUT>, <OPTION>, <AREA>
                  if (singletags.indexOf("~" + tag + "~") >= 0) {
                        hpEndElementEvent theEnd = new hpEndElementEvent
( tag );
                        theEnd.setDisplay(false);
                        abortparse = abortparse || EndElement ( theEnd
);
                  } else
                        tagstack.push(tag, false);
            } catch (Exception e) {
                  LogMessage("HTMLParser Error: " + e.toString() );
            }
      }

      private void EndATag( String tag ) {
            if (tag != null) {
                  try {
                        if (tagstack.toptag().equalsIgnoreCase(tag)) {
                              abortparse = abortparse || EndElement (
new hpEndElementEvent( tag ) );
                              tagstack.pop();
                        } else {
                              //Write out an end tag as character data
                              abortparse = abortparse ||
                                    CharacterData ( new
hpCharacterDataEvent( "</" + tag + ">" ) );
                              //Mark matching open tag to be popped
                              tagstack.match(tagstack.search(tag));
                        }
                        //Pop off all matched open tags
                        while (tagstack.topmatched()) {
                              hpEndElementEvent theEnd = new
hpEndElementEvent( tagstack.toptag() );
```

```
                                theEnd.setDisplay(false);
                                abortparse = abortparse || EndElement (
theEnd );
                                tagstack.pop();
                           }
/*                         if (tagstack.search(tag) > 0) {
                                while (!tagstack.empty() && !tag.equals
((String) tagstack.peek())) {
//System.out.println("POP " + (String) tagstack.peek() + " --- " + tag);
Comment(new hpCommentEvent("<!-- POP " + (String) tagstack.peek() +
" --- " + tag + "-->"));
                                    abortparse = abortparse ||
EndElement ( new hpEndElementEvent( (String) tagstack.pop() ) );
                                }
                                abortparse = abortparse || EndElement (
new hpEndElementEvent( (String) tagstack.pop() ) );
                                }
//else { System.out.println("SKIP " + tag); Comment(new hpCommentEvent
("<!-- SKIP " + tag + "-->")); }
*/
                    } catch (Exception e) {
                          LogMessage(e.toString());
                    }
               }
          }
}
```


JavaClasses.jar - Quick View Plus
File Edit View Archive Go Window Help
Name
Size
JavaClasses.jar
com
jclark
util
Hashtable$Enumerator.class 722
Hashtable.class 1,988
xml
apps
Doctype.class 6,117
Normalize.class 2,661
Time.class 1,208
output
SyncXMLWriter.class 2,388
UTF8XMLWriter$ReplacementTextOutputStream.class 986
UTF8XMLWriter.class 4,996
XMLWriter.class 806
parse
awt
Application.class 1,449
ApplicationImpl.class 1,791
Parser.class 463
ParserImpl.class 1,126
base
Application.class 1,403
ApplicationImpl.class 1,791
Parser.class 485
ParserImpl.class 1,036
io
Application.class 1,446
ApplicationImpl.class 1,787
Parser.class 432
ParserImpl.class 1,092
ApplicationException.class 395
AttributeDefinition.class 703
358
ZIP File
The contents of JavaClasses.jar


JavaClasses.jar - Quick View Plus
File Edit View Archive Go Window Help
Name
Size
CharacterDataEvent.class 358
CommentEvent.class 310
DocumentParser.class 736
DTD.class 696
ElementType.class 584
EndCdataSectionEvent.class 178
EndDocumentTypeDeclarationEvent.class 248
EndElementEvent.class 209
EndEntityReferenceEvent.class 184
EndPrologEvent.class 214
Entity.class 309
EntityManager.class 302
EntityManagerImpl.class 2,116
EntityParser$DeclState.class 1,437
EntityParser$DTDImpl.class 1,936
EntityParser$ElementTypeImpl$Attribute.class 1,072
EntityParser$ElementTypeImpl.class 2,571
EntityParser$EntityImpl.class 1,006
EntityParser$StartExternalSubsetEvent.class 470
EntityParser.class 23,392
LocatedEvent.class 225
MarkupDeclarationEvent.class 500
MessageId.class 1,322
Messages.class 2,774
NotWellFormedException.class 1,015
OpenEntity.class 840
ParseLocation.class 343
ParserBase.class 696
ProcessingInstructionEvent.class 360
StartCdataSectionEvent.class 182
StartDocumentTypeDeclarationEvent.class 252
StartElementEvent.class 556
StartEntityReferenceEvent.class 229
sax
7,350
ZIP File
The contents of JavaClasses.jar


Quick View Plus
File Edit View Archive Go Window Help
Name
Size
Driver.class 7,350
Locator.class 365
ReaderInputStream.class 1,043
WrapperException.class 396
tok
ASCIIEncoding.class 1,128
Buffer.class 1,406
ContentToken.class 1,719
EmptyTokenException.class 247
Encoding.class 23,300
EndOfPrologException.class 249
ExtensibleTokenException.class 353
InternalEncoding.class 1,113
InvalidTokenException.class 586
ISO8859_1Encoding.class 1,018
PartialCharException.class 362
PartialTokenException.class 251
Position.class 580
PrologParser.class 8,513
PrologSyntaxException.class 237
SingleByteEncoding.class 1,264
StringConversionCache$Bucket.class 624
StringConversionCache.class 2,516
TextDecl.class 2,630
Token.class 647
TokenException.class 223
UTF16BigEndianEncoding.class 1,028
UTF16LittleEndianEncoding.class 1,040
UTF8Encoding.class 2,829
XmlDecl.class 470
Version.class 201
META-INF
MANIFEST.MF 20,237
Main.class 5,844
457
ZIP File
The contents of JavaClasses.jar


JavaClasses.jar - Quick View Plus
File Edit View Archive Go Window Help
Name
Size
BugODB.class 457
BugODBObject.class 579
BugHTTPServer.class 662
Main$MenuItemListener.class 1,616
BugPanel.class 9,068
BugCanvas.class 6,109
HTTPThread.class 5,579
BasicHTTPServer.class 3,729
Header.class 1,451
BasicHTTPAction.class 503
BugChatEntryFrame.class 3,050
BasicHTTPClient.class 3,962
HTTPClientException.class 1,033
CGISimpleFile.class 3,086
BugAlert.class 1,878
ODBXMLStringParser.class 1,594
ODBStackObject.class 489
ODB.class 7,291
ODBXMLParser.class 4,482
ODBObject.class 3,043
ODBException.class 467
RPCResponseParser.class 5,887
RPCHandler.class 386
RPCRequestParser.class 4,822
RPCResponse.class 2,058
RPCRequest.class 1,366
CGIRPCDispatcher.class 3,139
CGIRPCDispatcher$RPCHandlerObj.class 665
RPCInstantMsgSender.class 2,429
RPCInstantMsg.class 3,436
Scheduler.class 2,038
CGIDumpMessages.class 2,948
CGIDeleteMessage.class 2,308
CGIIMForm.class 3,796
2,813
ZIP File
The contents of JavaClasses.jar

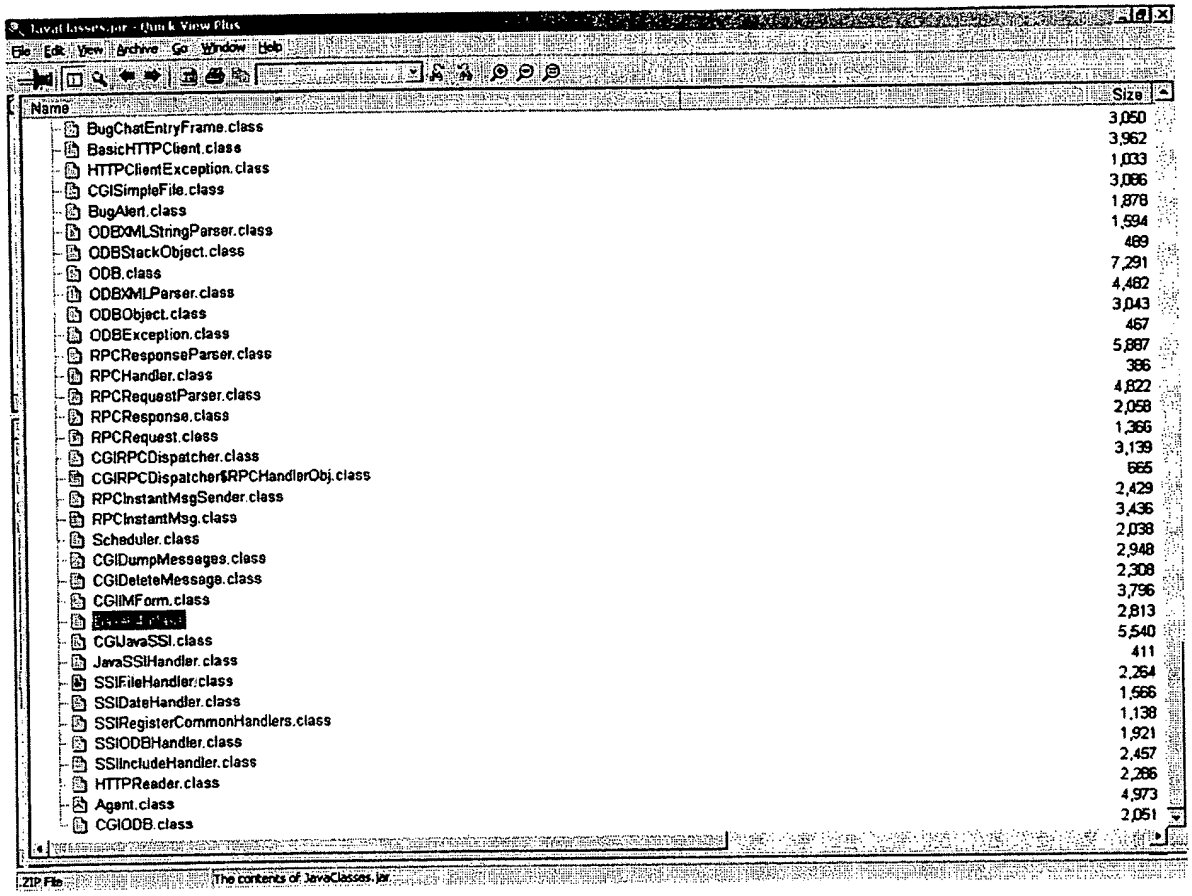

JavaClasses.jar - Quick View Plus
File Edit View Archive Go Window Help
Name
Size
BugChatEntryFrame.class 3,050
BasicHTTPClient.class 3,962
HTTPClientException.class 1,033
CGISimpleFile.class 3,086
BugAlert.class 1,878
ODBXMLStringParser.class 1,594
ODBStackObject.class 489
ODB.class 7,291
ODBXMLParser.class 4,482
ODBObject.class 3,043
ODBException.class 467
RPCResponseParser.class 5,887
RPCHandler.class 386
RPCRequestParser.class 4,822
RPCResponse.class 2,058
RPCRequest.class 1,366
CGIRPCDispatcher.class 3,139
CGIRPCDispatcher$RPCHandlerObj.class 665
RPCInstantMsgSender.class 2,429
RPCInstantMsg.class 3,436
Scheduler.class 2,038
CGIDumpMessages.class 2,948
CGIDeleteMessage.class 2,308
CGIIMForm.class 3,796
2,813
CGIJavaSSI.class 5,540
JavaSSIHandler.class 411
SSIFileHandler.class 2,264
SSIDateHandler.class 1,566
SSIRegisterCommonHandlers.class 1,138
SSIODBHandler.class 1,921
SSIIncludeHandler.class 2,457
HTTPReader.class 2,266
Agent.class 4,973
CGIODB.class 2,051
ZIP File
The contents of JavaClasses.jar

EXHIBIT D

Gossip Architecture [redacted]

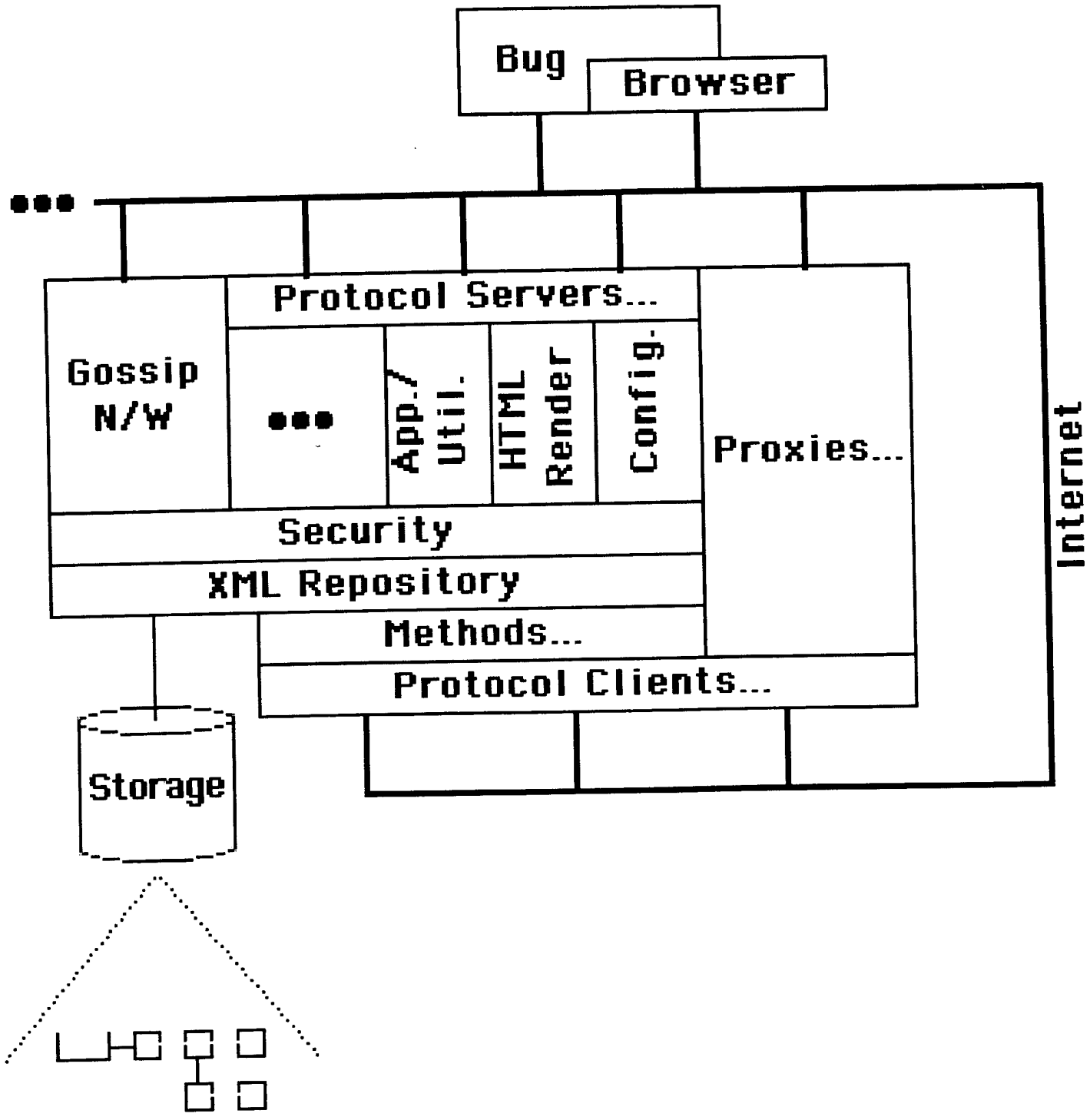


EXHIBIT E

Updated [redacted]

Headlines

updated [redacted]

Reuters - AP - AP U.S. - MSNBC

N.Korea says launched first satellite, not missile
North Korea announced on Friday it had successfully launched the country's first ever satellite on Monday, the day when Japan said the communist nation fired a missile over its territory.

Acting Russia PM sees economic dictatorhip
Acting Prime Minister Viktor Chernomyrdin outlined draconian measures to pull Russia out of crisis Friday, promising an ``economic dictatorship,'' a currency backed by gold and a shift away from reform policies.

Swissair salvage efforts to accelerate
With hopes of finding any survivors from the crash of a Swissair jet all but lost in the choppy and chilly Atlantic waters off Nova Scotia, crews were to accelerate salvage efforts Friday.

Swissair releases first list of Canada crash dead
Swissair said on Thursday that its special assistance team had notified the families of all but two of the 229 people killed in Wednesday night's crash off Nova Scotia.

Swissair Offers Flight, Cash To Victims' Families
Swissair launched an emergency plan it had hoped it would never use and offered families of people killed in the crash of Flight 111 a flight to Canada Friday and $20,000 in immediate aid for each passenger killed.

Bombing suspect alleges Pakistan bullied him-Post
Mohamed Sadeek Odeh, one of two suspects charged in the bombing of the U.S. Embassy in Kenya, claims Pakistani investigators refused to let him eat, drink or sleep for three days until he was pressured into a false confession, the Washington Post reported Friday.

Clinton in Dublin after euphoric N. Ireland tour
President Clinton meets leaders of the Irish republic on Friday, fresh from a whirlwind tour of Northern Ireland that seemed to lift the spirits of the people there as well as his own.

Clinton Senate ally attack's president's morals
Sen. Joseph Lieberman, a moderate Democrat from Connecticut, harshly rebuked President Clinton Thursday for his ``immoral'' behavior in the Monica Lewinsky scandal, opening the door

Stocks

Briefing

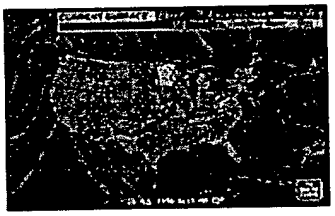
15:45 ET Dow -72, Nasdaq -13, S&P -6.08: Dow was off more than 200 points and has come roaring back in the past 20 minutes with the help of a large buy program.

Tech Tape

CPQ	-5/16	MSFT	-1 5/16	AMAT	-7/8
DELL	-2 3/16	NSCP	-1 5/8	CSCO	-1
IBM	+1 1/4	ORCL	-1 1/4	WDC	-9/16
SUNW	-1 5/16	INTC	+1 78	LU	-1 1/4
AOL	-7/8	MU	+15/16	YHOO	-2 9/16

Houston Weather

current weather as of [redacted], fog. temp 76. humidity 97. wind calm. visibility 3 miles.



Macintouch

[redacted]

Speed Doubler 8.1.1 offers compatibility with Mac OS 8.5; improved compatibilty with Mac Easy Open, FinderPop, StuffIt SpaceSaver, and others; a completely rewritten keyboard Power feature; and other fixes and enhancements.

Apple has posted Mac OS Runtime for Java 2.1 Early Access 2 (MRJ 2.1 EA2), an alpha version of an improved Java Virtual Machine based on version 1.1.6 of Sun's Java specification. Randy Wigginton, longtime Mac programmer and author of MacWrite, commented:

> The performance is somewhere between 4 and 6 times faster than MRJ 2.0! It is alpha, so don't commit any important data to it, but it is really cool to see my mac run java faster than my PC. ...I'm more excited about Apple these days than I have been since we shipped the original Mac.

for a new wave of criticism of the president from within his own party.

Bangladesh floods worsening, deaths at 580
The situation in Bangladesh, where the death toll from devastating floods rose by 28 to 580 on Friday, is getting worse every day, health officials said.

Ex-Rwanda PM jailed for life for genocide
Former Rwandan prime minister Jean Kambanda was sentenced Friday to life imprisonment by a United Nations court for his role in Rwanda's 1994 genocide.

Earlier Stories

- **North Korea says launches first satellite**
- **Minn. governor asks Clinton to stop Northwest strike**
- **Reno reviews Clinton probe of 1996 campaign finance**
- **Earl downgraded, lashes southeast U.S., three dead**
- **US immigrants face deportation for drunk driving**
- **Freeh expects more charges in U.S. embassy bombings**
- **136 Americans, AIDS Expert Die In Plane Crash**
- **U.S. Sees No Improvement In Maternal Death Rate**
- **Annan Mourns Air Crash Victims, Including U.N. Staff**
- **Press Group Urges Beijing To Release CBS Producer**
- **Clinton Sending Envoy Ross Back To Middle East**

Search News

Technology News

updated [redacted]

Reuters - Wired - ZDNet - AP

VLSI Says Industry Slowdown To Hurt Earnings
Specialty chip maker VLSI Technology Inc. said Wednesday its third quarter results would be lower than Wall Street expectations, due to an overall slowdown in the semiconductor industry.

According to a TechWeb article, Internet search services are unhappy with Sherlock, the new search feature planned for Mac OS 8.5. Their business model depends on getting Web surfers to their sites. When Sherlock does the searching, users get the benefits of their engines without opening their pages. Several companies say they are currently negotiating with Apple over the issue. (In Steve Jobs' preview of Sherlock during his Seybold keynote this week, banner ads from the search sites appeared in the bottom pane of the results window, just above a summary of the selected document. At one point during the demo, an ad for Windows 98 appeared on Jobs' Mac.)

iMacInTouch today reports on new drivers for several Epson printers. The drivers take advantage of yesterday's iMac Update 1.0.

Main Event Software yesterday released details about PhotoScripter, a new plug-in that makes it possible to control Adobe Photoshop via AppleScript. Demonstrated during Steve Jobs' Tuesday keynote address at the Seybold Seminars San Francisco/Publishing 98 conference and prominently featured in Apple's booth there, the plug-in addresses a longstanding need among publishing professionals who want to automate frequently performed operations. It will provide scripters with a huge dictionary covering most of Photoshop's options. Main Event plans a "preliminary" commercial release by the end of September or early October, with introductory pricing tentatively set at $299.

In yesterday's Seybold keynote Steve Ballmer, Microsoft's new president, pledged continuing support for the Mac but gloated over the inroads Windows has made into the publishing market. eMediaweekly has an extensive account.

Vimage yesterday reduced the price of its G3 upgrade cards for PCI Power Macs. The 300-MHz card is now $980, while the 233-MHz version has dropped to $449.

Sonnet Technologies yesterday announced the Sonata Pro 24, an accelerated graphics card for NuBus Macs. The $199.95 card, which Sonnet said is available immediately, is targeted particularly at Power Mac 7100 and 8100 owners who have to remove Apple's HPV or AV display card from their system's processor-direct slot to make room for a G3 CPU upgrade. As of this morning, details hadn't been posted at Sonnet's web site, but should be soon.

Terran Interactive will distribute VideoPrism*, a QuickTime 3 filter that enables videographers to apply both color correction and effects to their clips. The Mac version is slated to ship in the fourth quarter at an estimated street price of $159.

Another commercial CD-ROM, the Stock Illustration Source's SIS Nostalgia Collection, has been infected by the AutoStart worm, according to reader Chas

PSINet Builds Internet Service Presence In Japan
Commercial Internet service provider PSINet Inc. said it had completed its previously announced acquisition of Rimnet Corp., a leading Japanese commercial Internet service provider (ISP).

CMP Media Sees Q3 At $0.01-$0.02 Per Share
CMP Media Inc. said that based upon information currently available, it expects third quarter net income well below last year and in the range of $200,000 to $500,000, or $0.01 to $0.02 per share.

Computer Associates Seeks Latam Expansion
Computer Associates International Inc. Chairman and CEO Charles Wang said the company has opened up shop in Brazil as a part of a drive to expand businesses in Latin America.

BT Faces Struggle With Home Internet Product
British Telecommunications Plc faces a tough time selling its new BT Highway mass-market Internet access product and appears to have over-ambitious targets, experts said today.

Unicast Receives $7 Million From New Investors
Unicast Communications Corp., which makes tools to monitor advertising on the Internet, said Intel Corp., Grace Capital and several other investors have pumped $7 million worth of new capital into the company.

Telecom Operators In $1.5 billion Cable Project
A consortium of telecom operators signed a $1.5 billion project for a new fiber-optic cable link between Europe and the United States in order to cut waiting time on the World Wide Web.

Leading Internet Stocks Continue Rally
Leading Internet stocks came roaring back to life Wednesday as investors scooped up issues that suddenly looked like bargains after a storm of selling battered the sector over the past week.

Quark's Bid For Adobe Finds Little Support At Publishing Show
The hands said it all.

Onsale Passes 7 Million Bid Milestone
Onsale Inc. said it has received over 7 million online bids for product since launching its online auction over the World Wide Web in May 1995.

Earlier Stories

Belschner. The company recommended throwing away the disc and promised to send a replacement.

Al Guerra Enterprises, developer of an operating system for PowerPC-based Macs and clones that's based on Linux and Mach, is adding a PC emulator and a software library that provides a Macintosh-compatible environment. The projects are based on older versions of the two platforms, and they are "by no means finished," according to Guerra's statement, but source code is included with the latest pre-release Apokalypse CD.

Following yesterday's item about Symantec's JITspeed 3.0 just-in-time Java compiler for Netscape Navigator and Communicator 4.x, reader Richard Wallen reported problems with the software:

> With this software installed I had Netscape crash numerous times on Sun's Java demo page. Also it would not run the jmark test.
>
> MS IE has no trouble with any of the Sun Java demos. Also Netscape without the Symantec JIT also does not crash.

Symantec has also posted updated virus definition files for Symantec AntiVirus for Mac and Norton Antivirus for Mac.

Yesterday's report about game developer Interplay's decision to shut down its MacPlay division turns out to be old news: The company posted a statement about its plans several months ago. Reader Brehan Crawford notes that the company maintains "a message board for users to post questions about their current games, and to submit rants, flames, etc." Meanwhile, smaller Mac game developers are moving to fill the void, according to Nate Trost of Logicware:

> Although larger game publishers including Interplay/MacPlay have pulled out of the Macintosh game market, smaller Mac game companies do still exist! Because of the large company exodus, in late 1997 our company Logicware was forced to become not just a Mac game developer, but a publisher. Although it has been a battle to get ourselves on store shelves and let people know we exist with our limited resources, we're here and expanding. We actually licensed several MacPlay projects (Shattered Steel and Tempest 2000) which were never published, along with our own titles and titles from other companies. We will also be shipping the Interplay game Redneck Rampage as a Logicware Mac title by the end of the year. Hopefully, all these games will sell well, as we are

- **Oki, Compaq Team Up In Electronic Commerce System**
- **IBM Wins $3 Billion Cable and Wireless Contract**
- **Hitachi Sees Massive 98/99 Loss Amid Restructuring**
- **BellSouth New Brazil Cellular Unit Serving 535,000**
- **Government Seeks New Evidence From Microsoft**
- **Nasdaq Says Yahoo! Will Move Into Nasdaq 100**
- **Polaroid Says Restructuring On Track**
- **Motorola Sees Some Semiconductor Upturn**
- **Cyber Brokers Weather Double Volume With Ease**
- **America Online Makes eBay Its Online Auctioneer**
- **One-fourth Online To Use Broadband Access - Report**

Search News

DaveNet

[redacted]

DaveNet: Schwwwing!

James Allen Spahr has figured out how to script Adobe ImageReady from Frontier. This is a major major milestone!

Code Co-Op: Distributed version control for Windows.

InfoWorld interviews Jakob Nielsen.

Apple plays legal hardball with MacInTouch.

Wired: Quark gets in the XML loop.

NY Times: McGwire hits 54th after Sosa blasts 52nd.

Dilbert

already working on new licenses to try and bring more cool games to the Mac in 99!

PageMark 1.0.2 is a $5 shareware extension that flashes a bar in the window area when you scrolled to the end of a page.

Version 2.0b3 of Snak, Kent Sorensen's full-featured IRC Client for the Mac, offers improved scriptability among other enhancements.

Our Recent News and special reports also cover the following topics, among others:

1. **Jobs Seybold keynote and announcements**
2. Trojan Horse story
3. AutoStart worm
4. G3 dim video